المدرسة الوطنية العليا في الأمن السيبراني
NATIONAL SCHOOL OF CYBERSECURITY

1ST YEAR BASIC TRAINING IN CYBER SECURITY

INTRODUCTION TO OPERATING SYSTEMS 1 (SYST1)

Dr. Sassi BENTRAD
✉ : sassi.bentrad.enscs@gmail.com | | sassi.bentrad@enscs.edu.dz
LISCO Laboratory (**L**aboratoire d'**I**ngénierie des **S**ystèmes **CO**mplexes) / University of Badji Mokhtar-Annaba (**UBMA**)
National **S**chool of **C**yber **S**ecurity (**NSCS**)

Basic Training in Cyber Security (1BT)

Formation de Base en Cyber-Sécurité (1FB)

Chapter

Files and Directories Management

SYST1'2025/2026

Module Description Sheet : *Introduction to Operating Systems 1 (SYST1)*

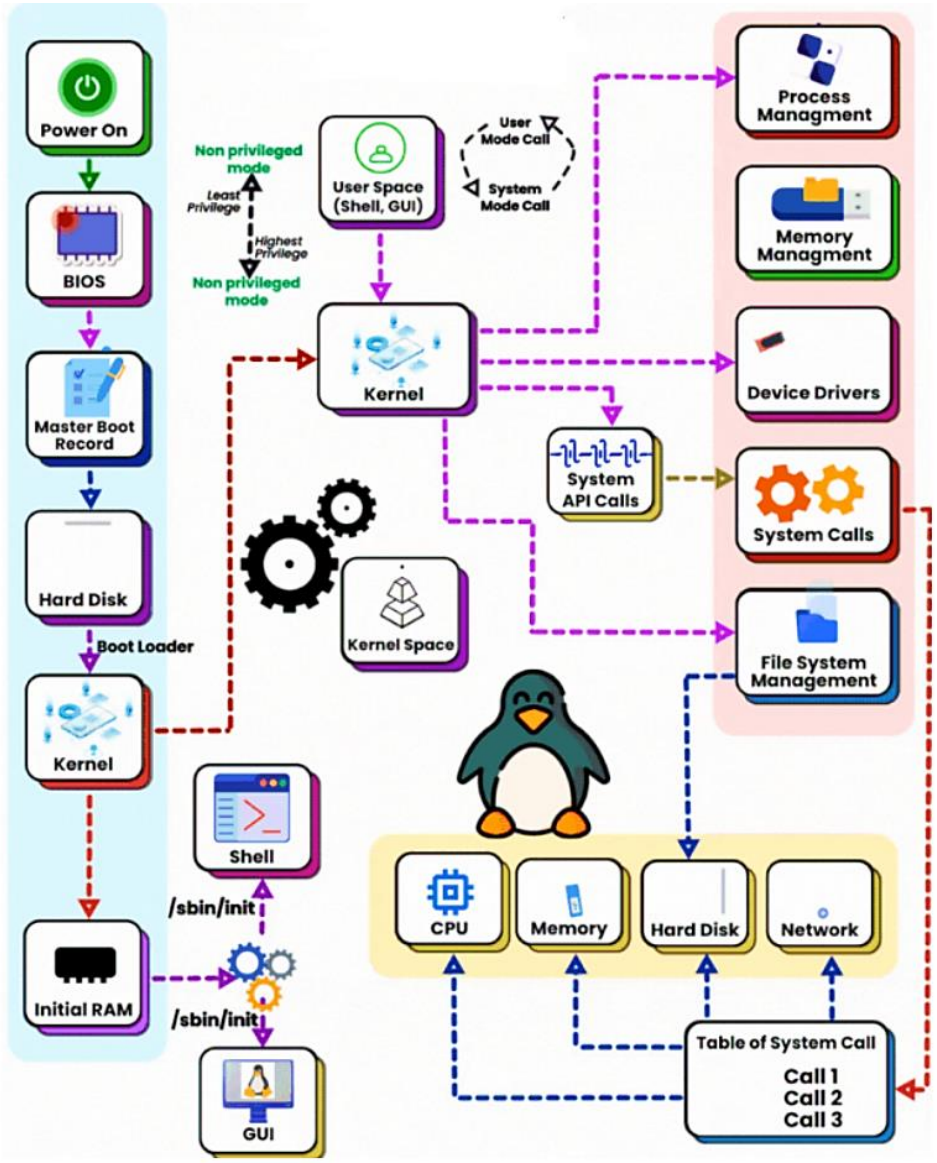
Files and Directories Management (10 %)

Course Content

CHAPTER 5

- Introduction
- The File Management System
- Linux File System
  - *File Tree Structure*
  - *Top-Level Directories*
  - *Special directories*
  - *File System Utilities*
  - *File Access: Absolute Path & Relative Path*
- Files & Directories
  - *Nomenclature of files and directories*
  - *Links (Physical & Symbolic)*
  - *Special Characters*
- Basic Commands
  - *File operations*
  - *Wildcard*
  - *Regular Expression*
  - *Links (Physical & Symbolic)*
- Advanced Commands

❑ How Linux Works ?



❑ The Chaos of Early Linux Filesystems ...

- In the early days of **Linux**, file organization lacked consistency and standardization.
- **Each distribution** (e.g., *Slackware*, *RedHat*, *Debian*) had its **own directory layout**, causing confusion among users and developers.
- **Configuration files**, binaries, and libraries were often stored in **different locations**, depending on the distribution.
- This inconsistency made **system administration, scripting, and software packaging** unnecessarily complex.
- To restore order, the **Filesystem Hierarchy Standard (FHS)** was introduced — providing a **unified structure** and clear conventions for all Linux systems.

❑ File Management System — *FMS, a Core Component of the OS*

Main Roles of the File Management System

1. File Manipulation

- Enables applications and users to **create**, **open**, **read**, **write**, **rename**, and **delete** files.
- Provides a **programmatic interface** (system calls) for file operations.

2. Storage Space Allocation

- Manages **allocation of blocks** (fixed-size units) in secondary memory.
- Each file is given a **variable number of blocks**, depending on its size.
- Ensures **efficient disk utilization** and prevents fragmentation.

❑ File Management System — *FMS, a Core Component of the OS*

shelby

Main Roles of the File Management System

3. File Location and Metadata

- Maintains **information describing each file** (name, owner, size, permissions, timestamps, etc.).
- Uses **inodes (index nodes)** to locate and identify data on disk.
- Provides a **mapping between logical file structure and physical storage**.

4. File Security and Access Control

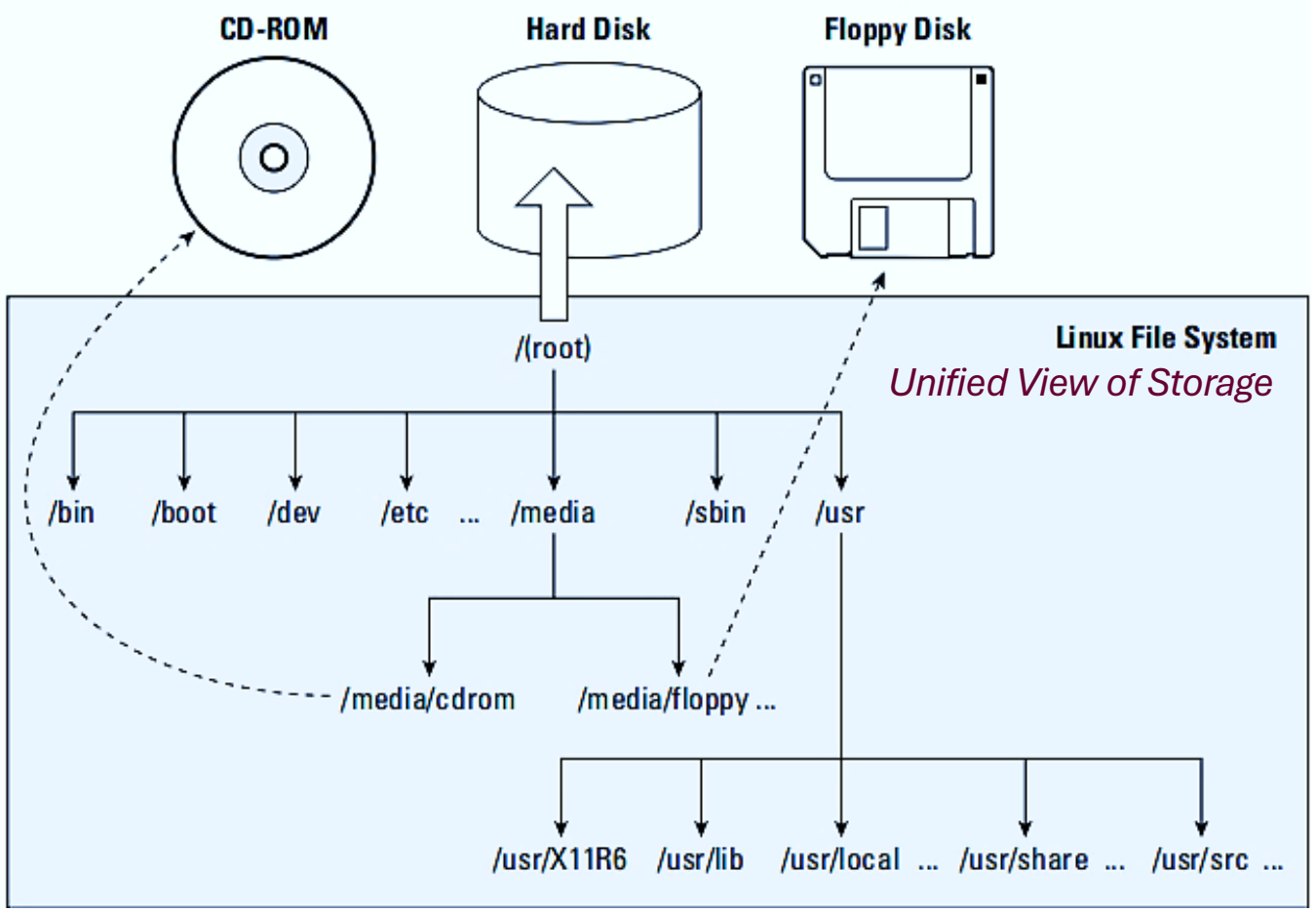
- Ensures **controlled sharing** of files among users and programs.
- Applies **permissions**, **ownership**, and **access control lists (ACLs)** to protect data.
- Supports **isolation** between processes and users.

💡 **Key Idea :** The **FMS** acts as the **bridge** between user-level file operations and low-level disk management — ensuring **organization, efficiency, and security** within the operating system.

❑ The concept of the Linux File System

The Figure illustrates the concept of the **Linux file system** (which is **the same in any Linux OS**):

- ✓ **Unified and Hierarchical Structure**
- ✓ **Spanning Multiple Physical Devices**
- ✓ **Transparency and Flexibility**



💡 **Key Takeaway :**
The **Linux filesystem** is a **unified namespace** — a **single tree** that can seamlessly span **local, removable, and network storage**.

❑ The concept of the Linux File System

- **A Well-Structured Hierarchy**

  The **Linux file system** is organized into a **tree-like hierarchy** with a clearly defined set of **top-level directories**.

  Each directory under the **root (/)** serves a **specific system purpose** — such as storing executables, configuration files, user data, or temporary files.

  This structure ensures **consistency and predictability** across all Linux distributions.

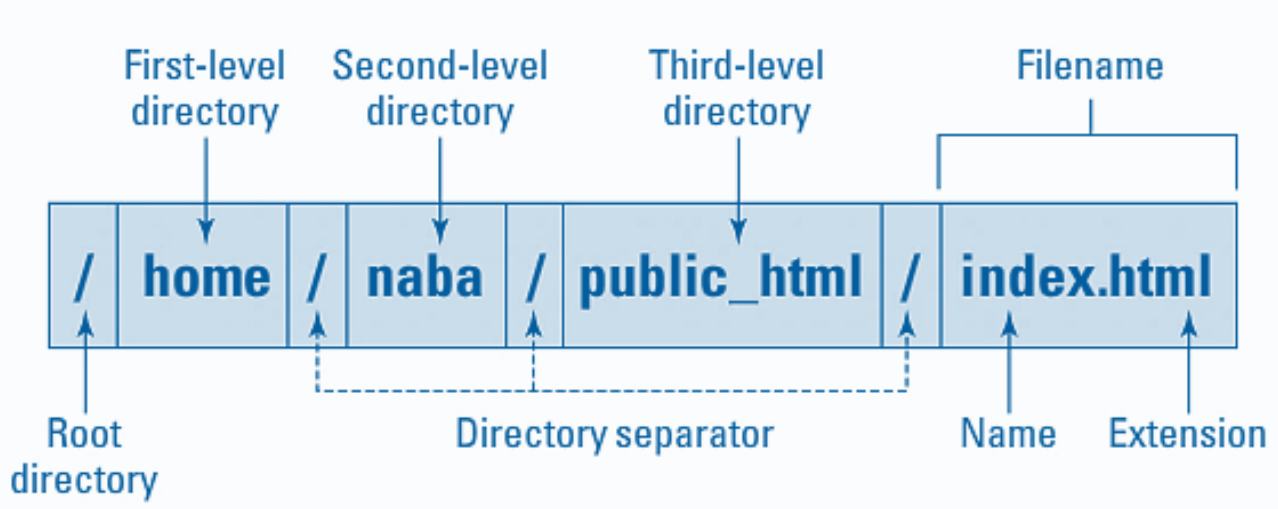
- **Navigating the File System**

  Knowing **the role of each directory** helps you quickly locate files and understand how the system is organized.

  When facing a **new Linux environment**, you can easily **infer where to look** for configuration files, executables, or logs.

  This knowledge is essential for **system administration**, **troubleshooting**, and **scripting**.

The path name of a file shows the sequence of directories leading up to the file.



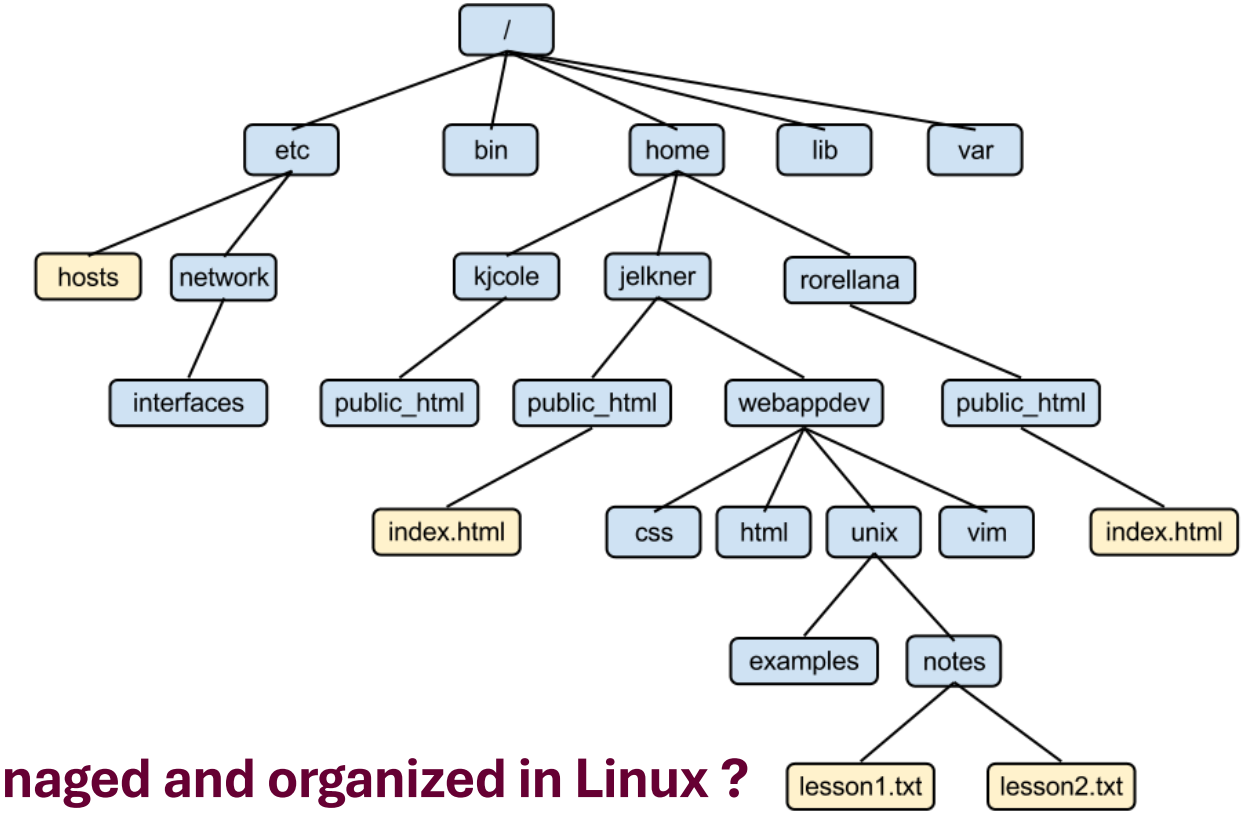
❑ File System (FS) ?

Files: are used to store data.

There are several **types of files**:

- ✓ *Text ;*
- ✓ *Image ;*
- ✓ *Audio ;*
- ✓ *Script ;*
- ✓ *Database; etc.*

Directories: are used for organization.



How are files and directories managed and organized in Linux ?

The File System (FS) ?

- **FS** defines the organization of data on a storage medium.
- **FS** in Linux describes a **directory and subdirectory tree**, starting from "**/**" a called the **root.**
- **Linux FS** is **unique, does not depend on storage space**.

❑ File System (FS) ?

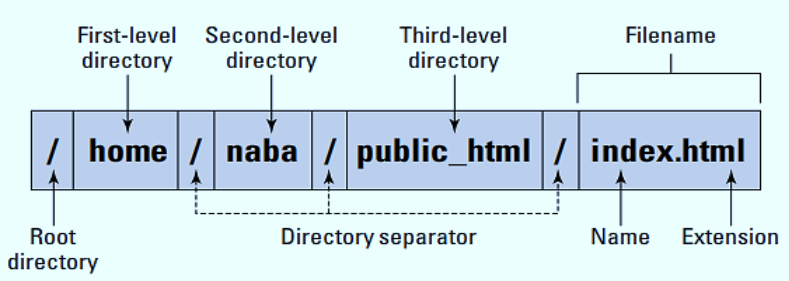
Files with a name that can have a **maximum number of characters** ranging from **14** to **255** depending on the type of file system used. The **Extension System** is not managed by **Linux** but by **services** and **applications**.

Extension	Service or Application	Extension	Service or Application
`.c`	Source file in language C	`.jpeg, .jpg, .gif, .png`	Image files
`.cpp`	Source file in language C++	`.txt`	Text file
`.h`	Header file in C or C++	`.pdf`	Adobe Acrobat Reader file
`.java`	Source file in java language	`.htm ou .html`	Web Pages Web in HTML langage
`.class`	Compiled java class	`.php`	Script in php language
`.o`	Compile-generated object file	`.ps`	Postscript Files
`.a`	Static Library	`.tar`	Archive in tar format
`.so`	Dynamic Library	`.tgz`	Archive in tar format comp. by gzip
`.sh`	Script shell	`.deb`	Debian Package (Installer)
`.gz ou .z`	File compressed using gzip	`.rpm`	Paquetage RedHat
`.bz2`	File compressed using bzip2		

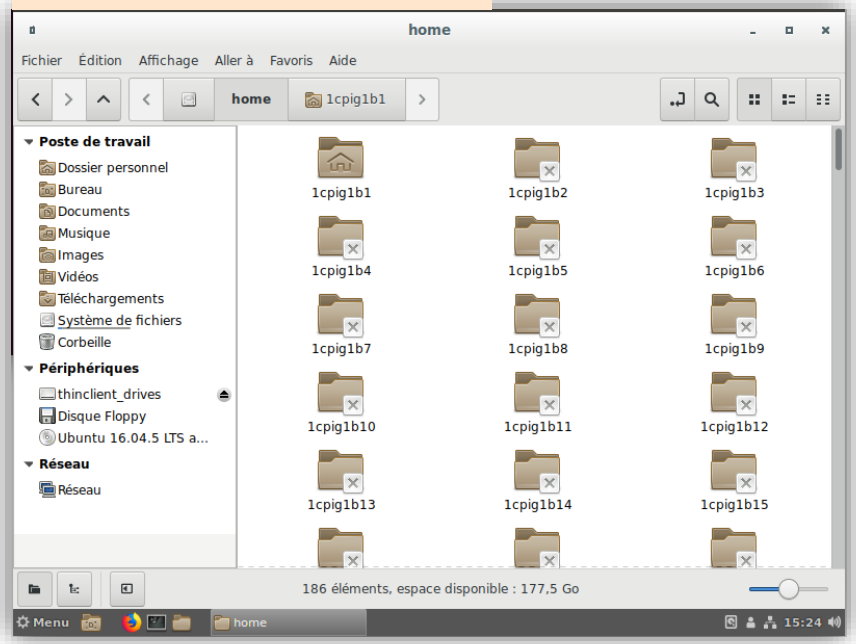
File System (FS) : Personal Directory

Personal directory: The Linux system allocates each user a **home directory** with the user's name in the `'/home/'` directory.

Each user can create files and directories in their home directory `'/home/user_name'`

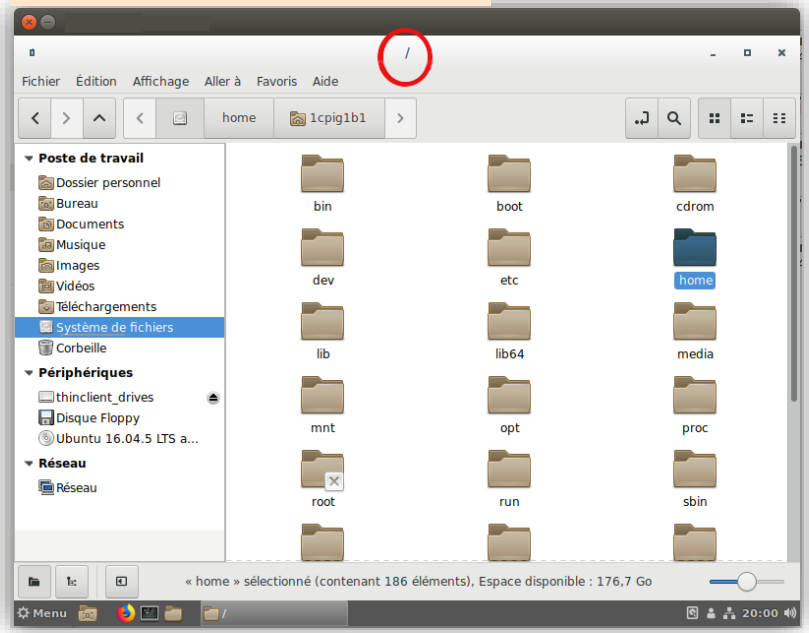


Directory : /home/



/	.	..	~
slash	dot	double dot	tilde
Root Directory	**Working Directory**	**Parent Directory**	**Home Directory**

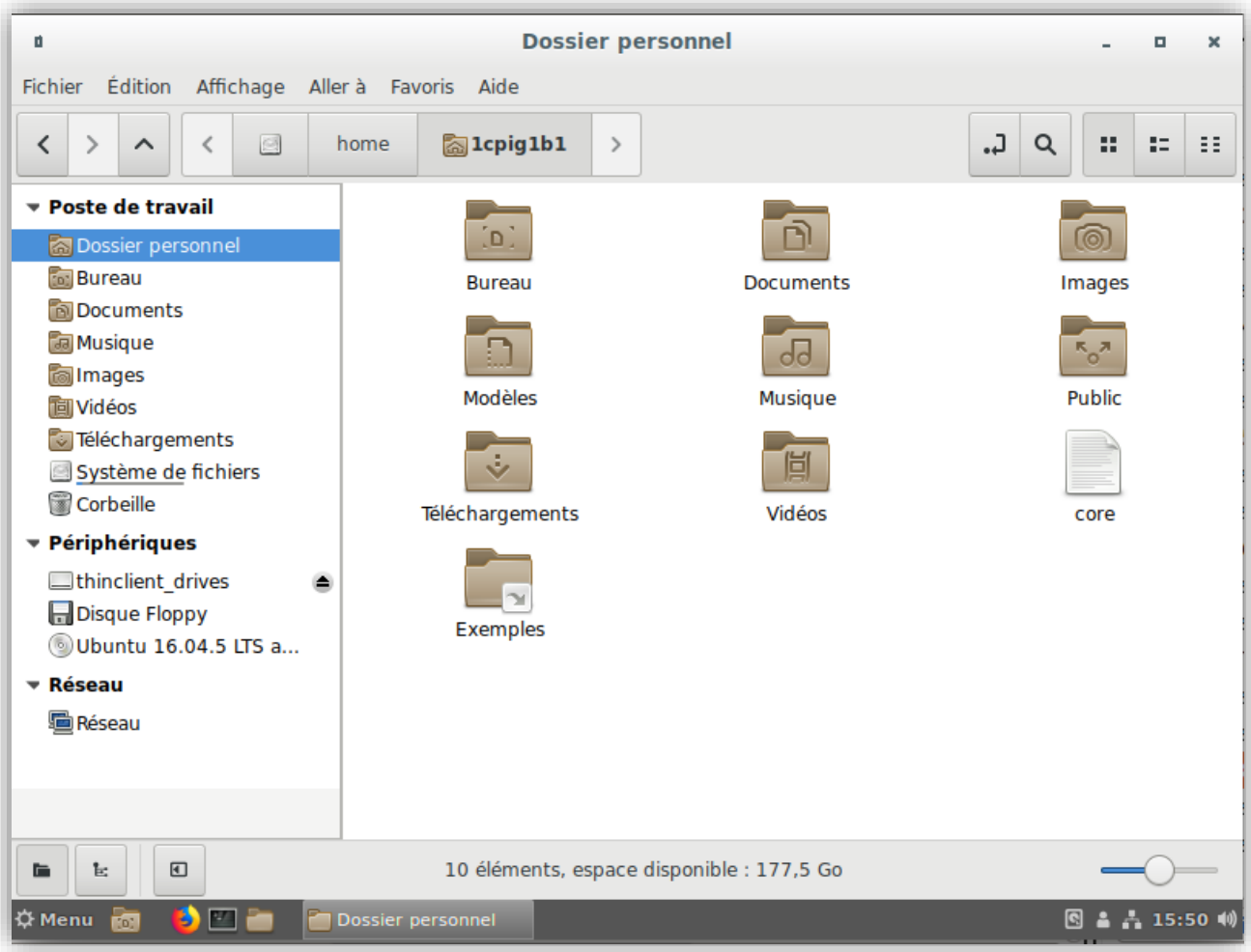
Root Directory : /



❑ File System (FS) : Home Directory

Home directory : `/home/1cpig1b1/`

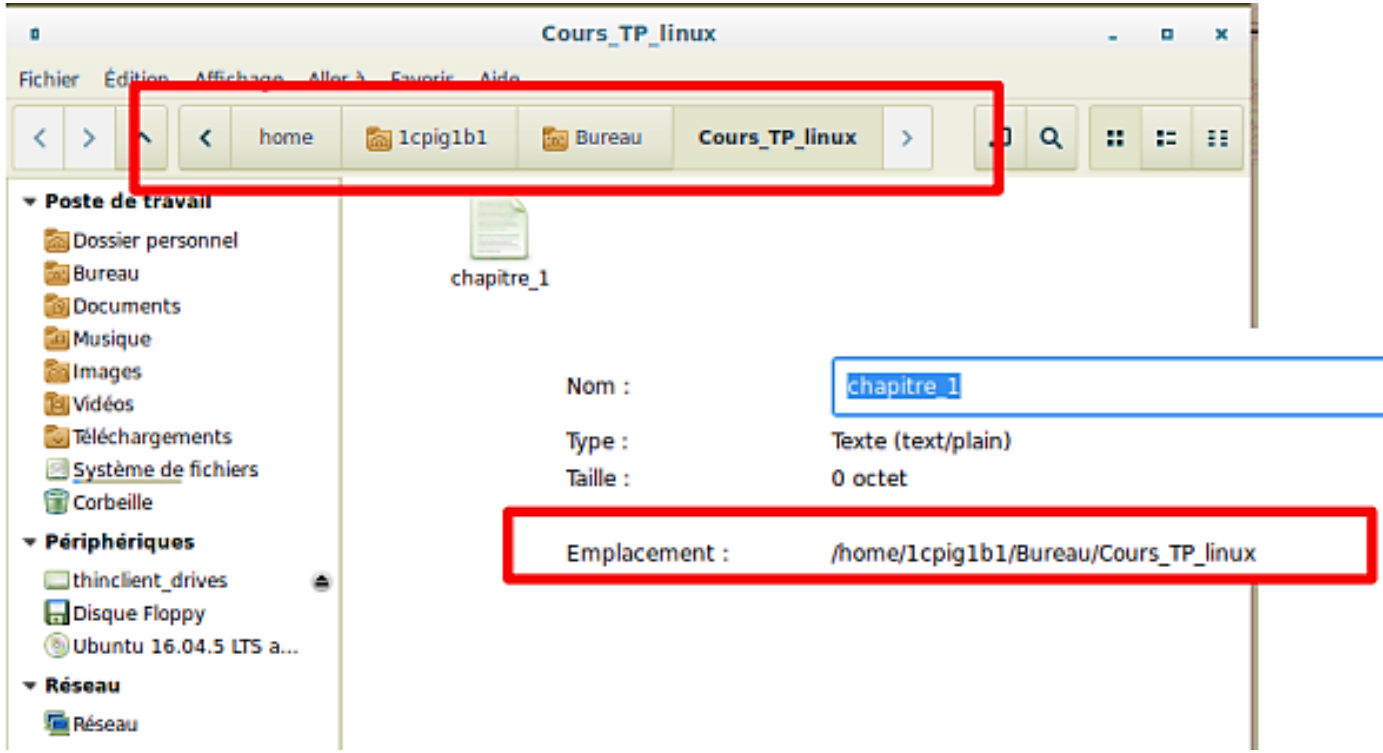
/	.	..	~
slash	dot	double dot	tilde
Root Directory	**Working Directory**	**Parent Directory**	**Home Directory**



File System (FS) : Paths

The paths : the **Path** is used to set the location of a file or directory in the system.

An absolute or **full path** is the list of directories and subdirectories separated by the character '/' from the root '/' to the object (file or directory).



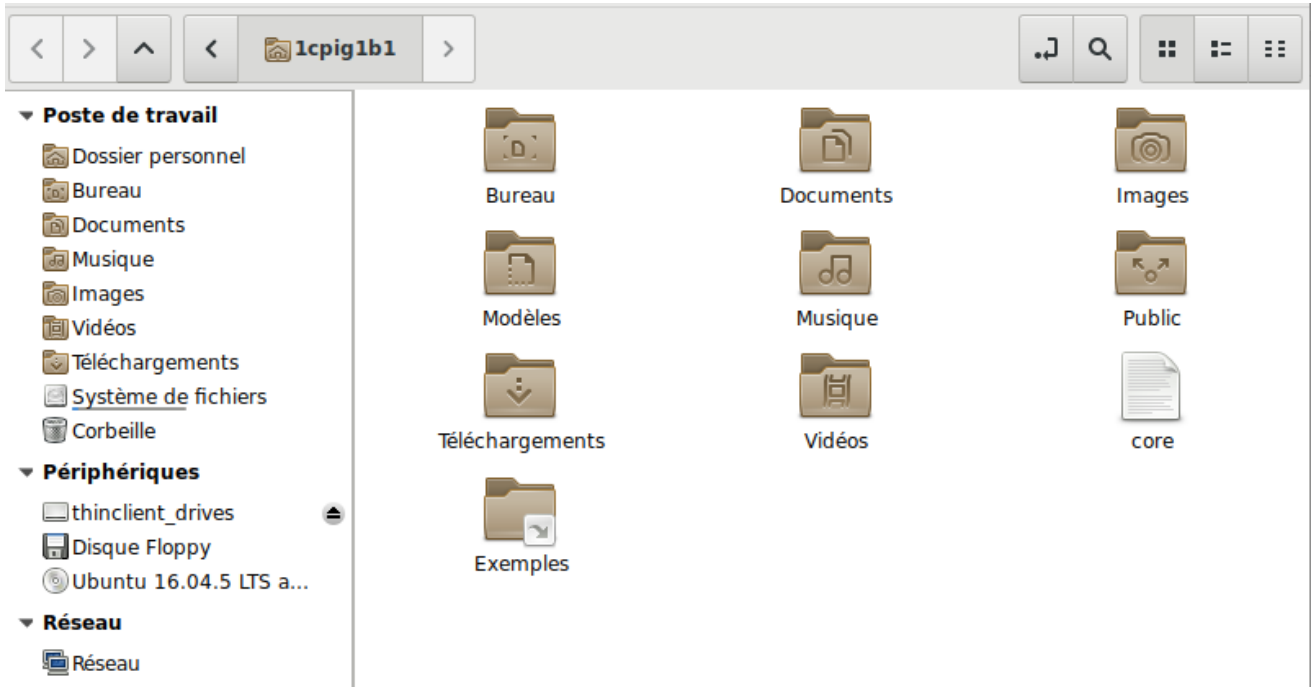
Example:
If the teacher shares a chapter with the user `1cpig1b1`, the user can access the chapter with the absolute path :

```
/home/1cpig1b1/Bureau/Cours_TP_linux/chapitre_1
```

File System (FS) : Paths

A **relative path** is the list of directories and subdirectories separated by the '/' character from the **current** or **active directory** to the object (file or directory)

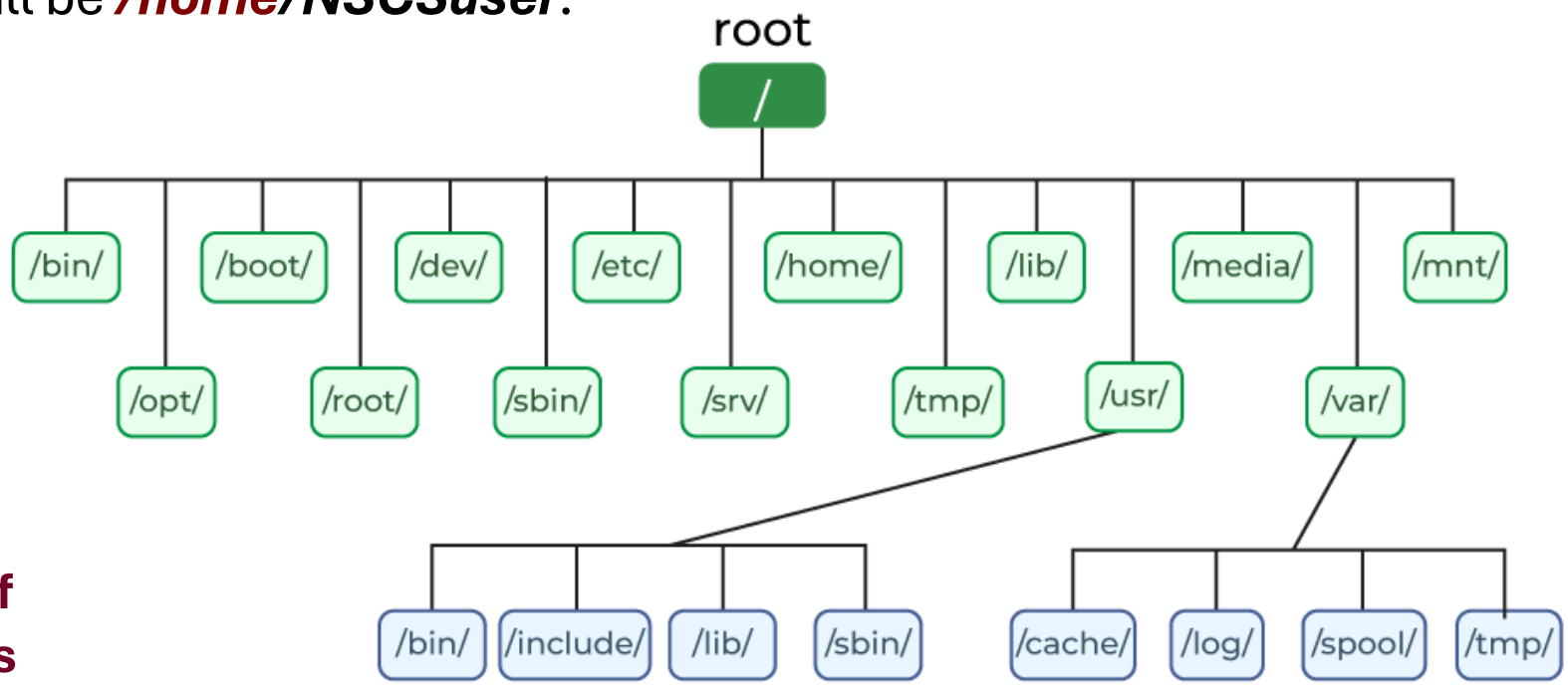
Example: In the past example, if the user `1cpig1b1` tells the teacher I am in `/home/1cpig1b1/` the teacher to avoid repetition will just give him the relative path with respect to his active directory `Bureau/Cours_TP_linux/chapitre_1`



The concatenation of **current directory path** and **relative path** gives **absolute path.**

The UNIX/Linux File System

- On **UNIX/Linux OS**, everything is organized as **files**. **There is no C: type drive like on Windows**.
- The file system starts from the **root directory /** and **branches** out into various **subdirectories**.
- The folder where users store their documents is **/home**. If your login is ***NSCSuser***, then your personal file will be ***/home/NSCSuser***.

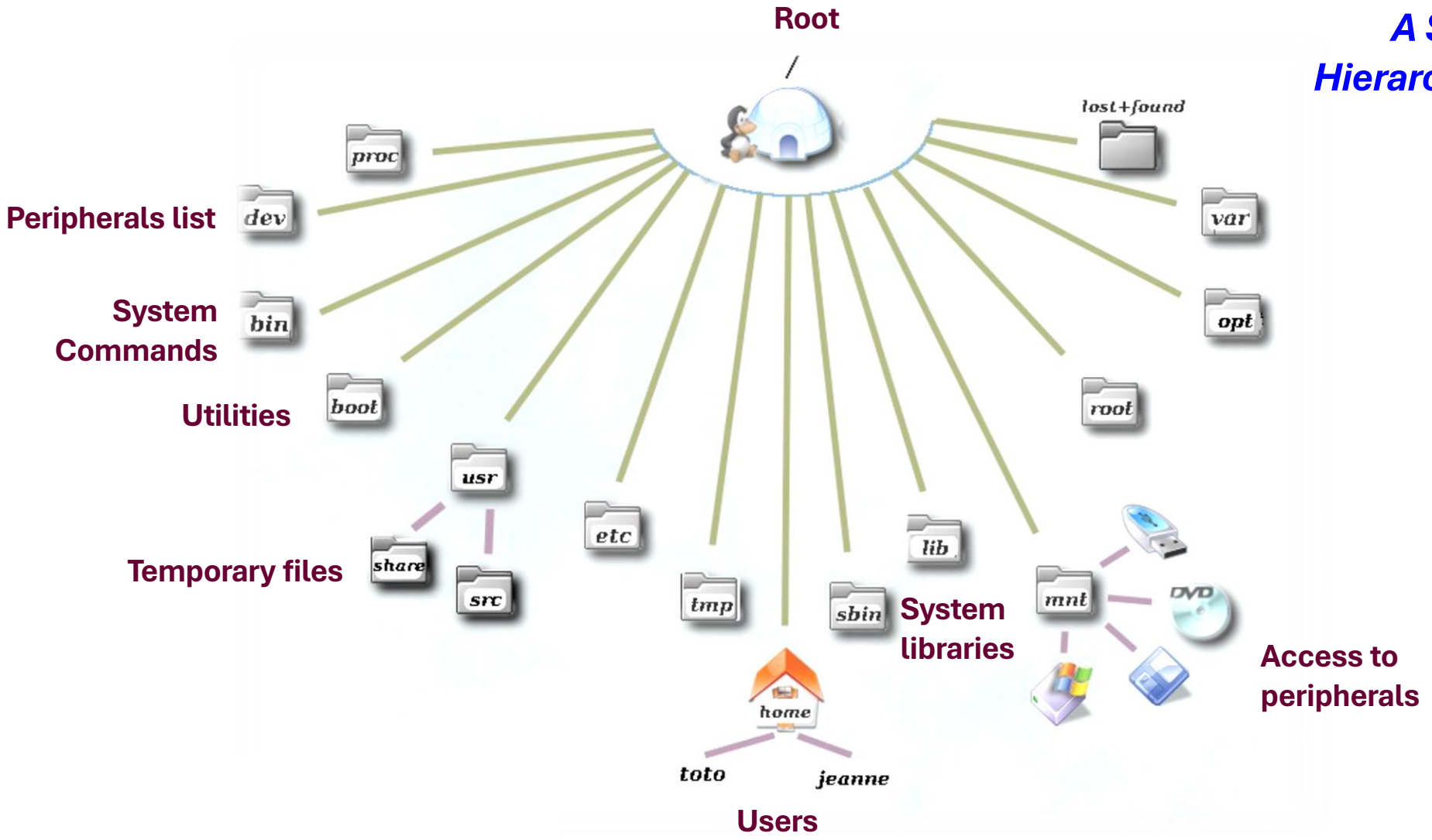


The structure of folders and files

❑ The UNIX/Linux File System

- On **UNIX/Linux**, any element is represented as a **file**.
- **4 File types:**
  - ✓ **Regular files (*Ordinary*)**: Contain data, program(text, binary, etc.).
  - ✓ **Directories**: Contain other files or directories.
  - ✓ **Symbolic links**: Pointers to other files or directories.
  - ✓ **Special files**:
    - **Device files**: Represent hardware devices (e.g., **/dev/sda**).
    - **Named pipes and sockets**: Used for inter-process communication.
- Each file is characterized by its ***name***, ***size***, ***access rights***, ***owner***, ***dates of creation***, ***modification***, etc.
- **File tree structure**

The UNIX/Linux File Tree Structure

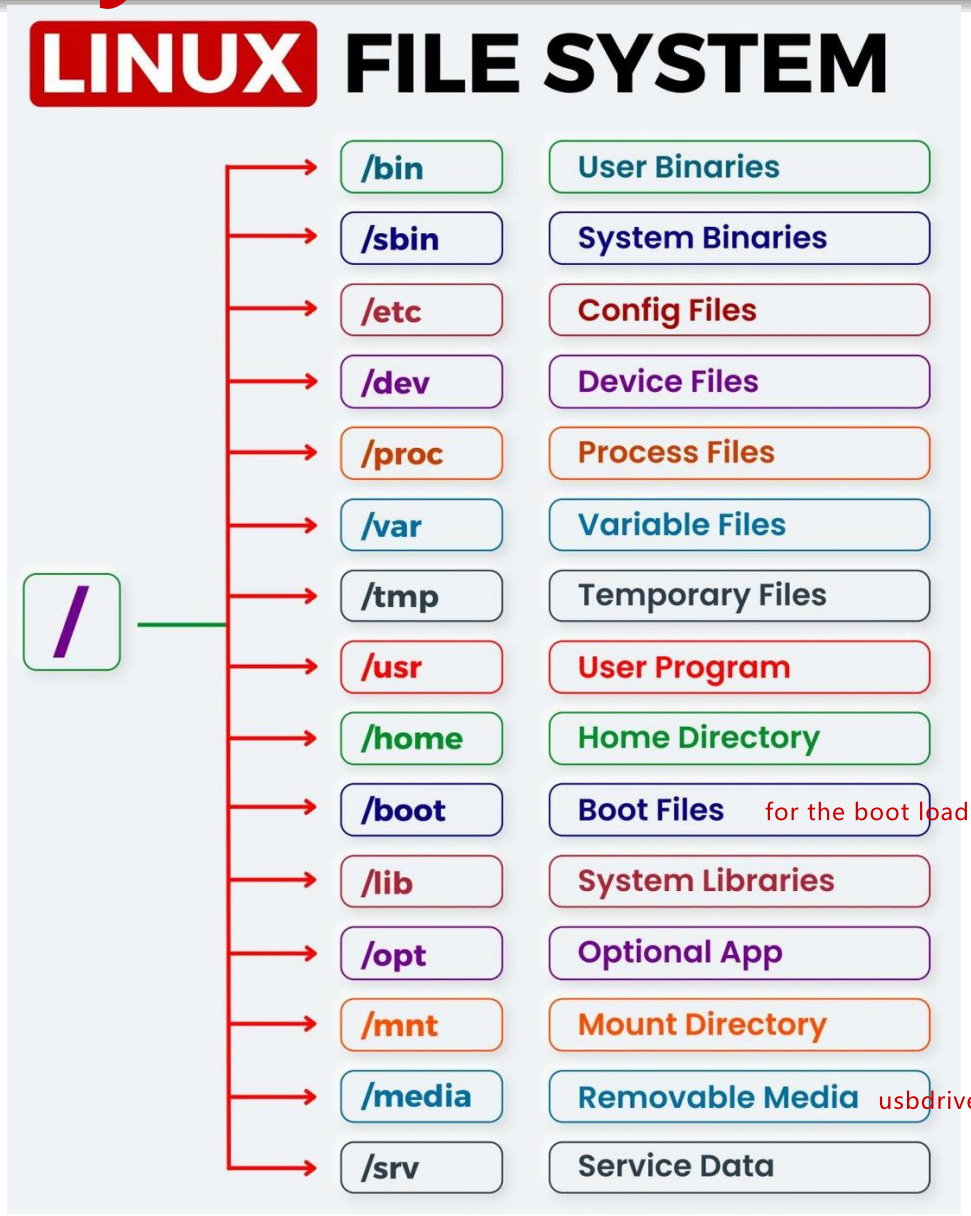


The UNIX/Linux File Tree Structure

It is important to understand a basic Linux directory structure as you may use some of the directories or files already prepared by **Linux OS**.

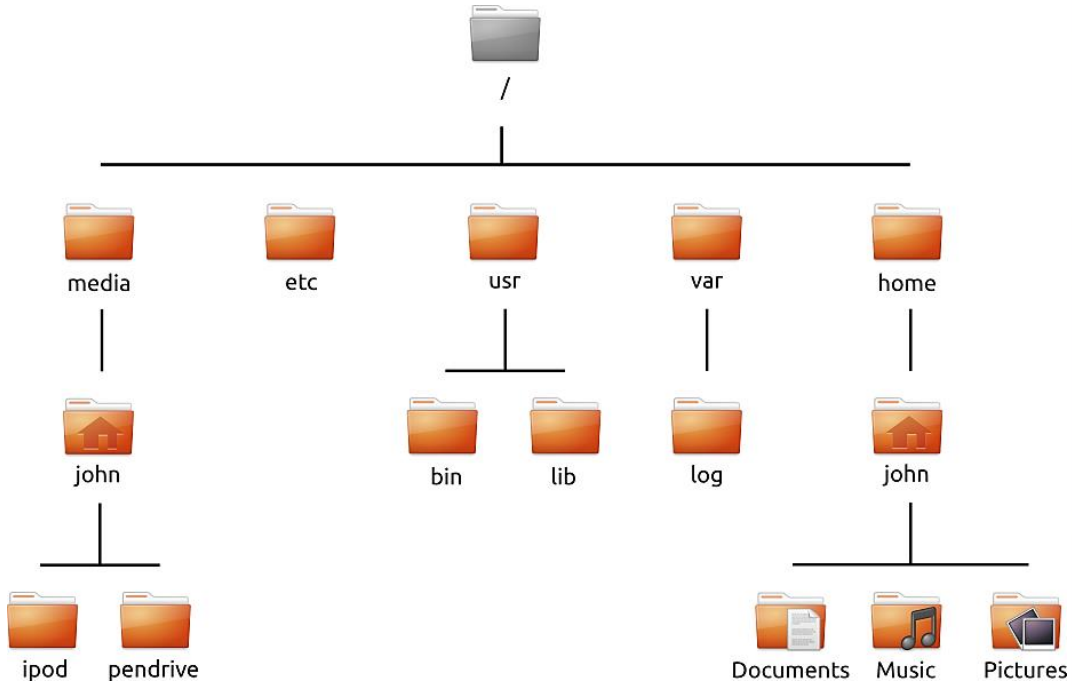
The **Linux directory structure** starts with the **root** (**/**) directory.

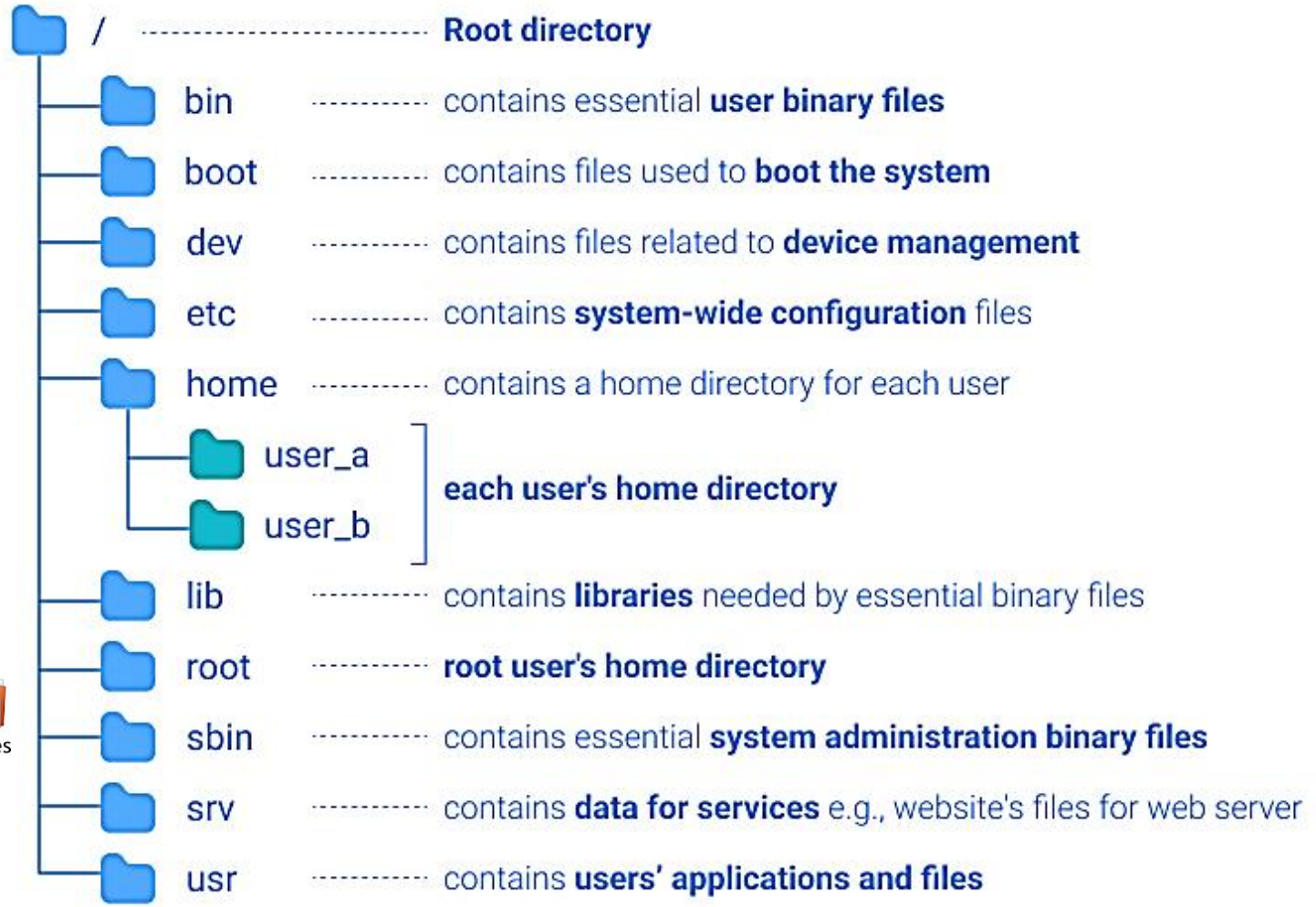
Usually, there are around **10-20 directories** under the root directory but **the structure can differ by distribution.**



💡 **Key Takeaway :** The UNIX/Linux file tree is a **unified, hierarchical structure** where everything — files, directories, and devices — fits neatly under a **single root /**.

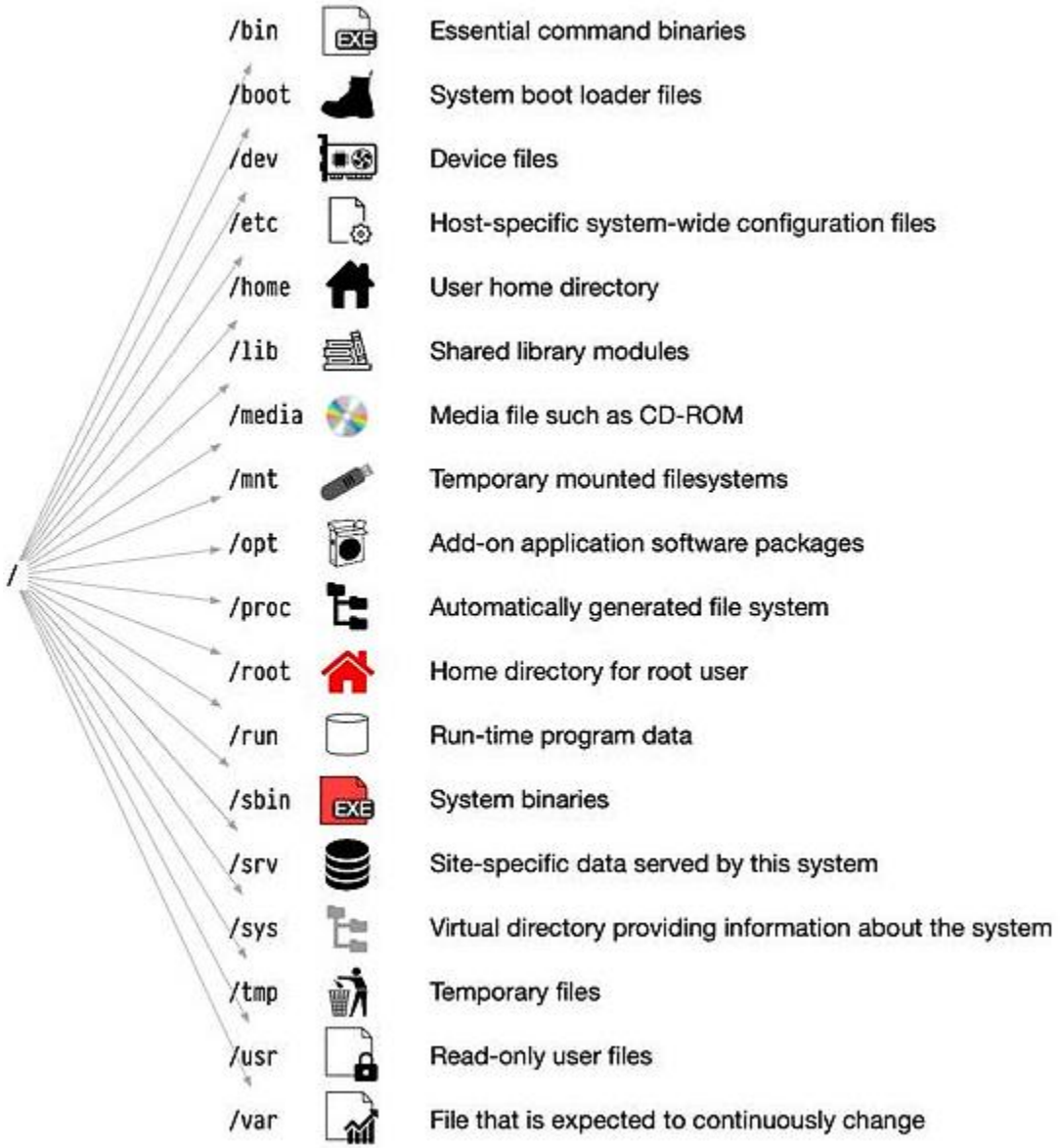
❑ The UNIX/Linux File Tree Structure

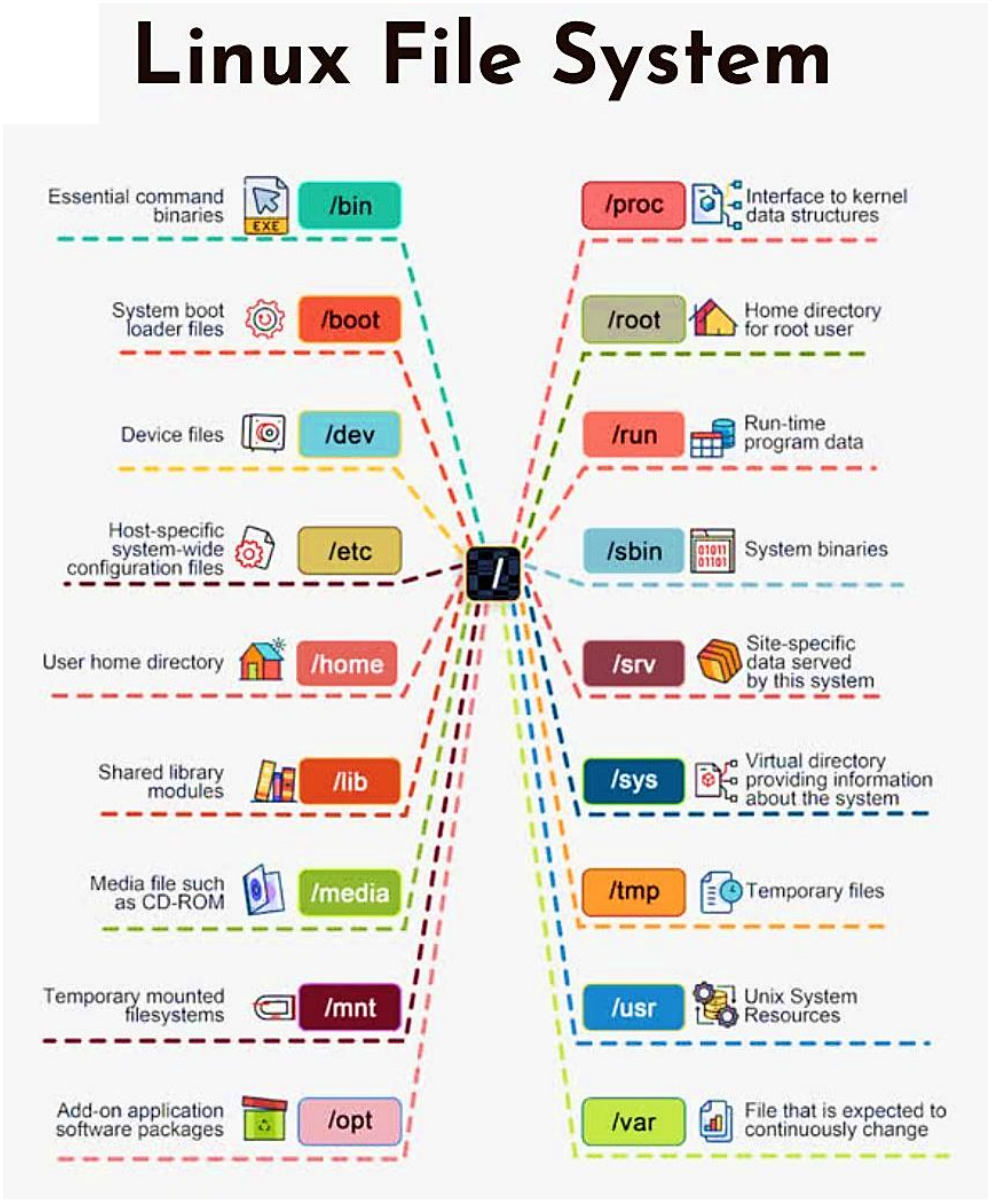




Linux Directory Structure

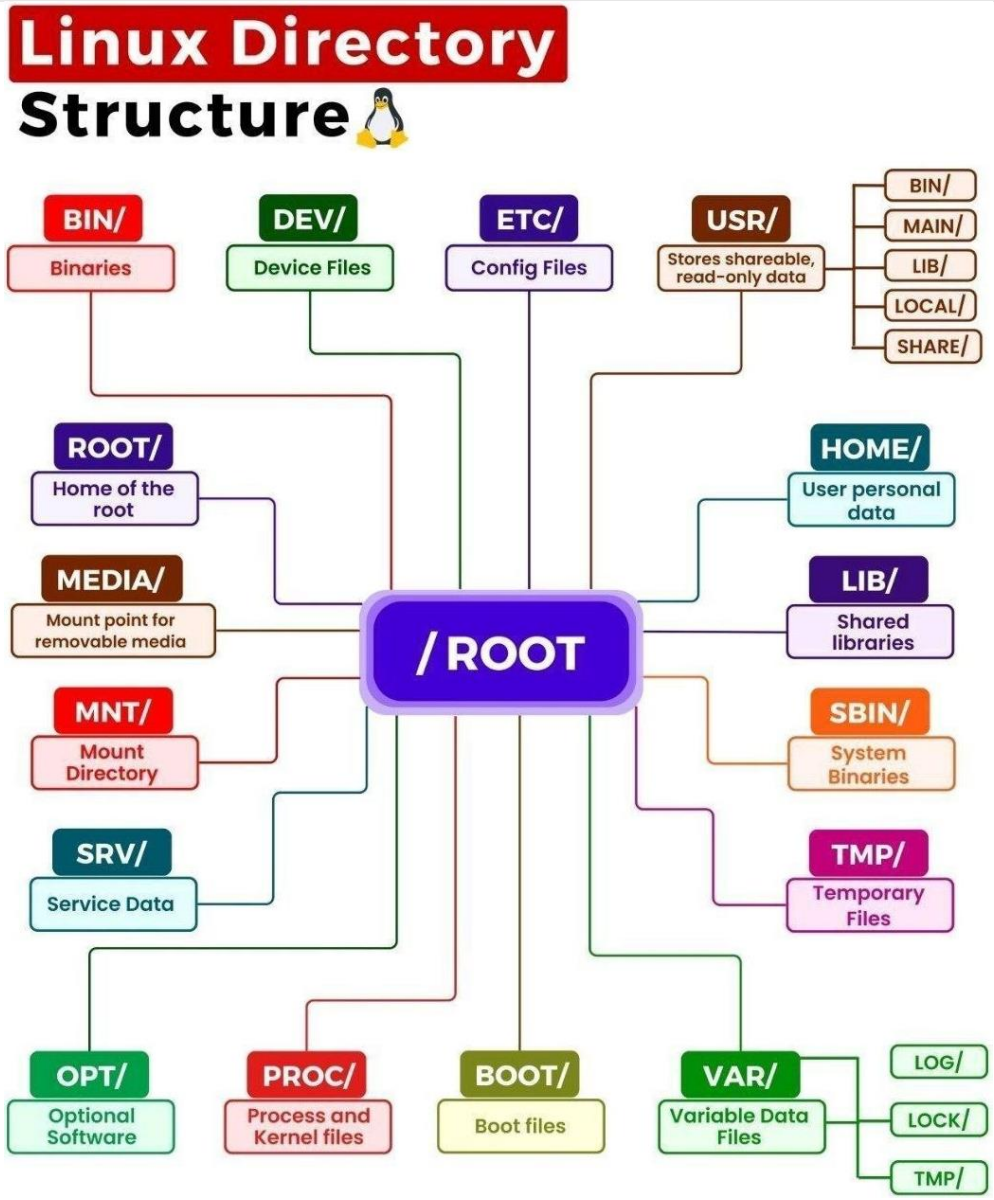
❖ THE FILE SYSTEM





❑ The UNIX/Linux File Tree Structure

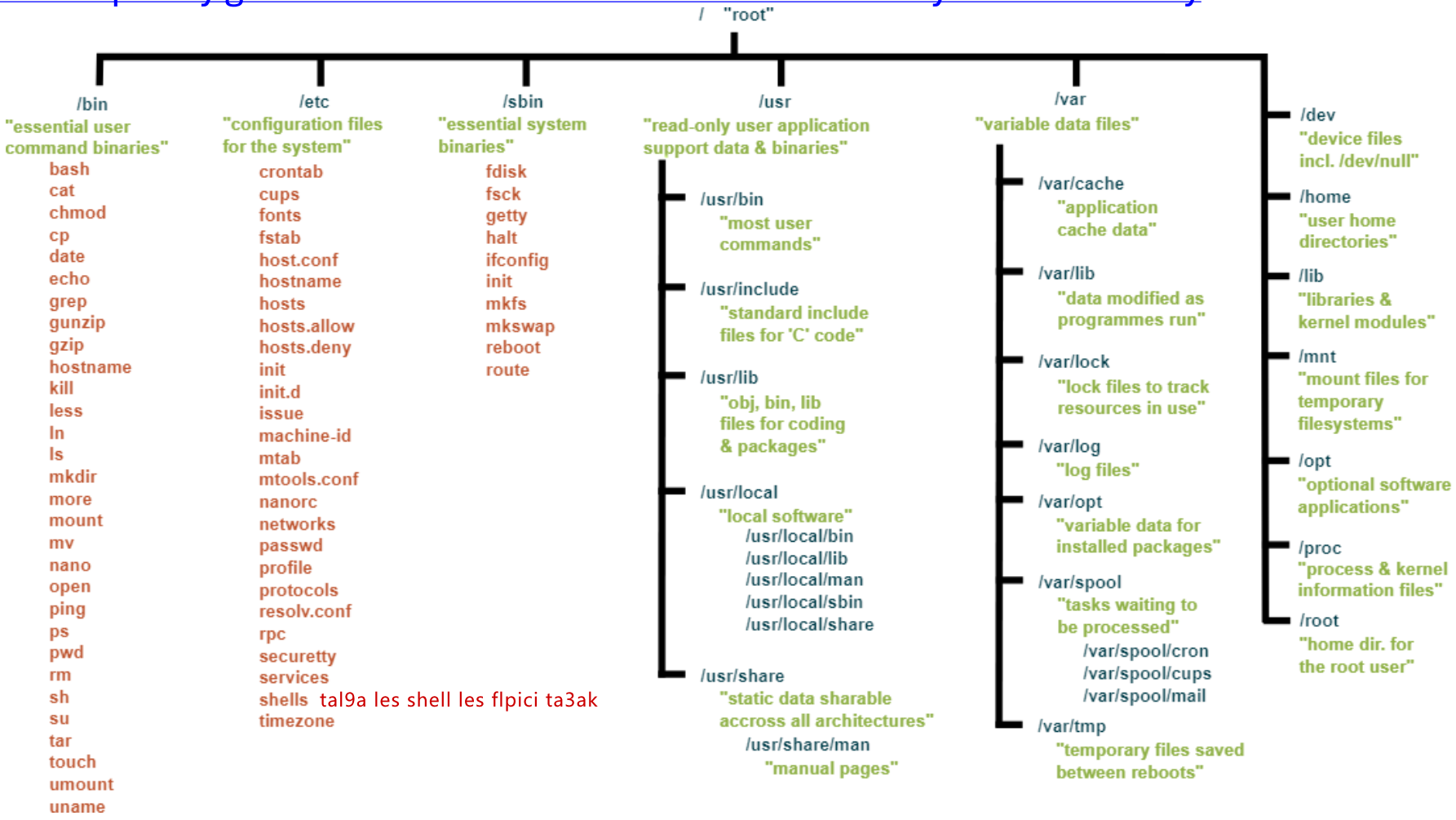
- The Linux file system has **a well-defined set of top-level directories**, and some of these directories have **specific purposes**.
- Finding your way around the file system is easier if you know the **purpose of these directories.**
- You also become adept at guessing where to **look for specific types of files** when you face a new situation.



❖ THE FILE SYSTEM

The following is an overview of a **standard Unix filesystem**. The exact hierarchy depends on the platform. Your file/directory structure may differ slightly.

Source: https://swcarpentry.github.io/shell-novice/reference.html#filesystem-hierarchy



❑ Top-Level Directories in the Linux File System

N. Barkakati,
Linux All-in-One Desk Reference For Dummies,
2nd ed. Hoboken, NJ, USA: Wiley Publishing, Inc., 2006.

Top-Level Directories in the Linux File System

Directory	*Description*
/	This `root` directory forms the base of the file system. All files and directories are contained logically in the `root` directory, regardless of their physical locations.
/bin	Contains the executable programs that are part of the Linux operating system. Many Linux commands, such as `cat`, `cp`, `ls`, `more`, and `tar`, are located in `/bin`.
/boot	Contains the Linux kernel and other files that the LILO and GRUB boot managers need. (The kernel and other files can be anywhere, but placing them in the `/boot` directory is customary.)
/dev	Contains special files that represent devices attached to the system
/etc	Contains most system configuration files and the initialization scripts (in the `/etc/rc.d` subdirectory)
/home	Conventional location of the home directories of all users. User naba's home directory, for example, is `/home/naba`.
/lib	Contains library files for all programs stored in `/sbin` and `/bin` directories (including the loadable driver modules) needed to start Linux
/lost+found	Directory for lost files. Every disk partition has a `lost+found` directory.
/media	A directory for mounting file systems on removable media, such as CD/DVD-ROM drives, floppy disks, and Zip drives. Contains the `/media/floppy` directory for mounting floppy disks and the `/media/cdrom` or `/media/cdrom0` directory for mounting the CD/DVD-ROM drive. If you have a CD/DVD recorder, you find a `/media/cdrecorder` directory instead of `/media/cdrom`.

❑ Top-Level Directories in the Linux File System

N. Barkakati,
Linux All-in-One Desk Reference For Dummies,
2nd ed. Hoboken, NJ, USA: Wiley Publishing, Inc., 2006.

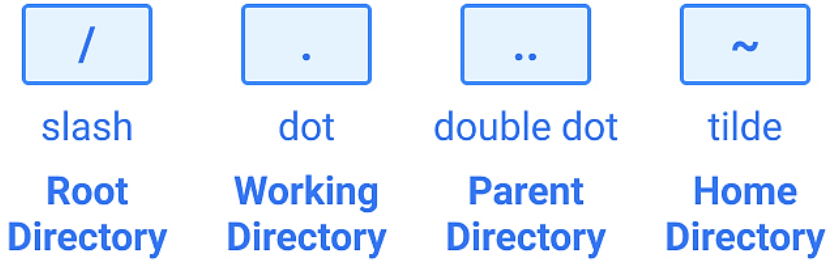
Directory	*Description*
`/mnt`	A directory for temporarily mounted file systems
`/opt`	Provides a storage area for large application software packages. For example, some distributions install the OpenOffice.org office suite in the `/opt` directory.
`/proc`	A special directory that contains various information about the processes running in the Linux system
`/root`	The home directory for the `root` user
`/sbin`	Contains executable files representing commands typically used for system-administration tasks and used by the `root` user. Commands such as `halt` and `shutdown` reside in the `/sbin` directory.
`/srv`	Contains data for services (such as Web and FTP) offered by this system
`/sys`	A special directory that contains information about the devices, as seen by the Linux kernel
`/tmp`	A temporary directory that any user can use as a *scratch* directory, meaning that the contents of this directory are considered unimportant and usually are deleted every time the system boots
`/usr`	Contains the subdirectories for many important programs, such as the X Window System (in the `/usr/X11R6` directory) and the online manual. (Table 3-2 shows some of the standard subdirectories in `/usr`.)
`/var`	Contains various system files (such as logs), as well as directories for holding other information, such as files for the Web server and anonymous FTP server

❑ Top-Level Directories in the Linux File System

N. Barkakati,
Linux All-in-One Desk Reference For Dummies,
2nd ed. Hoboken, NJ, USA: Wiley Publishing, Inc., 2006.

Subdirectory	*Description*
`/usr/lib`	Contains the libraries for C and C++ programming languages; also contains many other libraries, such as database libraries, graphical toolkit libraries, and so on
`/usr/local`	Contains local files. The `/usr/local/bin` directory, for example, is supposed to be the location for any executable program developed on your system.
`/usr/sbin`	Contains many administrative commands, such as commands for electronic mail and networking
`/usr/share`	Contains shared data, such as default configuration files and images for many applications. For example, `/usr/share/gnome` contains various shared files for the GNOME desktop, and `/usr/share/doc` has the documentation files for many Linux applications (such as the `bash` shell, the Sawfish window manager, and The GIMP image-processing program).
`/usr/share/man`	Contains the online manual (which you can read by using the `man` command)
`/usr/src`	Contains the source code for the Linux kernel (the core operating system)

❑ Special directories :



❑ Hidden files :

On **UNIX/Linux OS**, hidden files start with a period. For example, **~/.bashrc** is a hidden file, in the user's home directory, that contains the configuration of his shell.

❑ Wildcards : ? and *

The characters **?** and ***** in file and directory names are used to represent any character. '**?**' represents a single character, while '*****' represents any number.

❑ File System Utilities

Linux offers powerful **command-line tools** to manage disks, partitions, and file systems effectively. These utilities are essential for **system setup, maintenance, and recovery**.

Key Commands and Their Functions

- **fdisk / parted** – Create, view, and manage **disk partitions**.
  Used to define how a physical disk is divided into logical sections.
- **mkfs – Make File System**: formats a partition with a chosen file system type (e.g., **ext4**, **xfs**, **vfat**).
- **fsck – File System Check**: scans and **repairs file system errors** to maintain data integrity.

  *Always unmount a file system before running **fsck** or resizing it to avoid data corruption.*
- **resize2fs – Resize ext-based file systems** (expand or shrink without data loss).

❑ File System Utilities

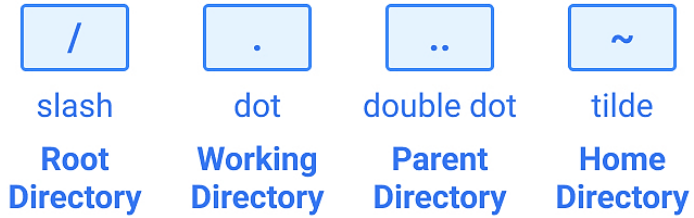
- **Types of File Systems** : **ext4** (*Fourth Extended File System*), **Btrfs**, **XFS**, **FAT32**, **NTFS** ;
- **Mounting and Unmounting File Systems** (Commands such as : **mount**, **unmount**);
- **File Permissions and Ownership :**
  - Each file has three types of permissions: **Read** (**r**), **Write** (**w**), **Execute** (**x**);
  - User and Group Ownership: The owner and the group have specific permissions.
  - Managing file permissions and ownership : commands such as **chmod**, **chown**, and **chgrp**.

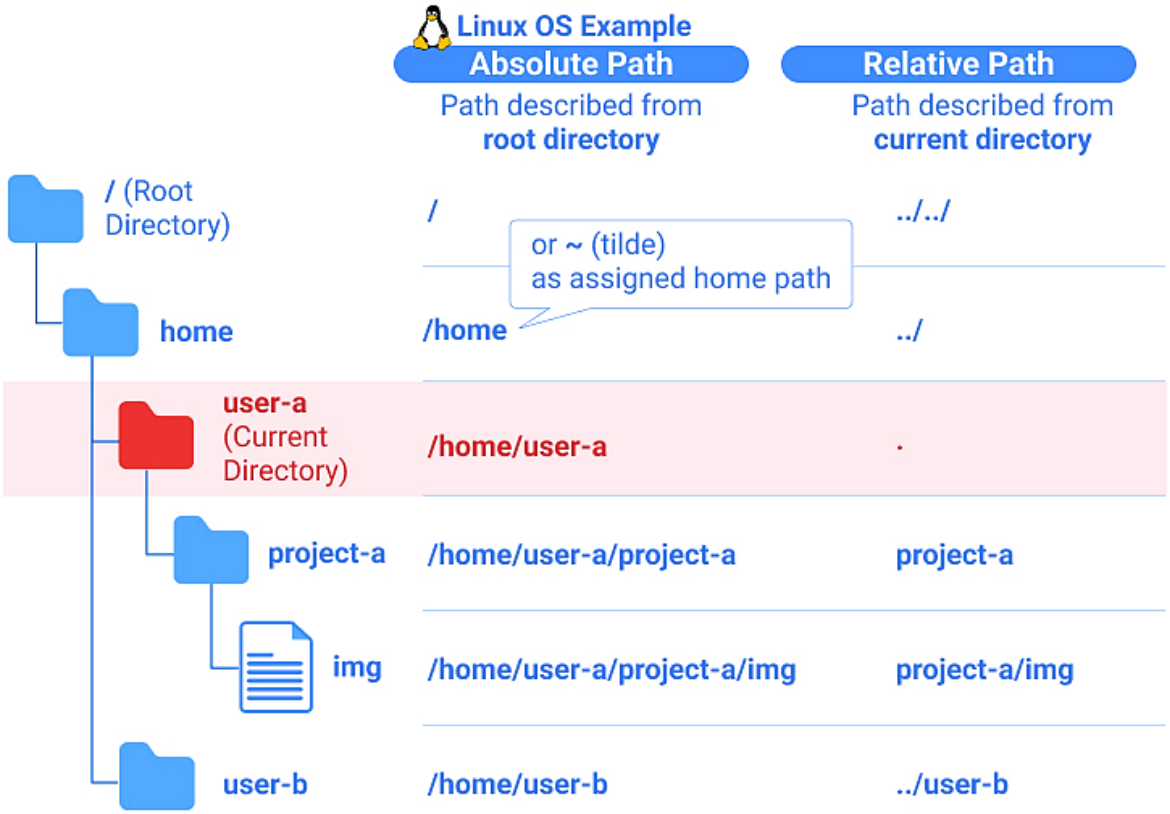
File Access: Absolute Path and Relative Path

There are two approaches to writing a file path - **absolute path** and **relative path**.

The **absolute path** structure can be different by OS. The main figure illustrates the absolute path structure on Linux OS.

Understanding these concepts is very important for setting up links to files properly. The key concepts are shown in the illustration below.





❑ File Access: Absolute Path and Relative Path

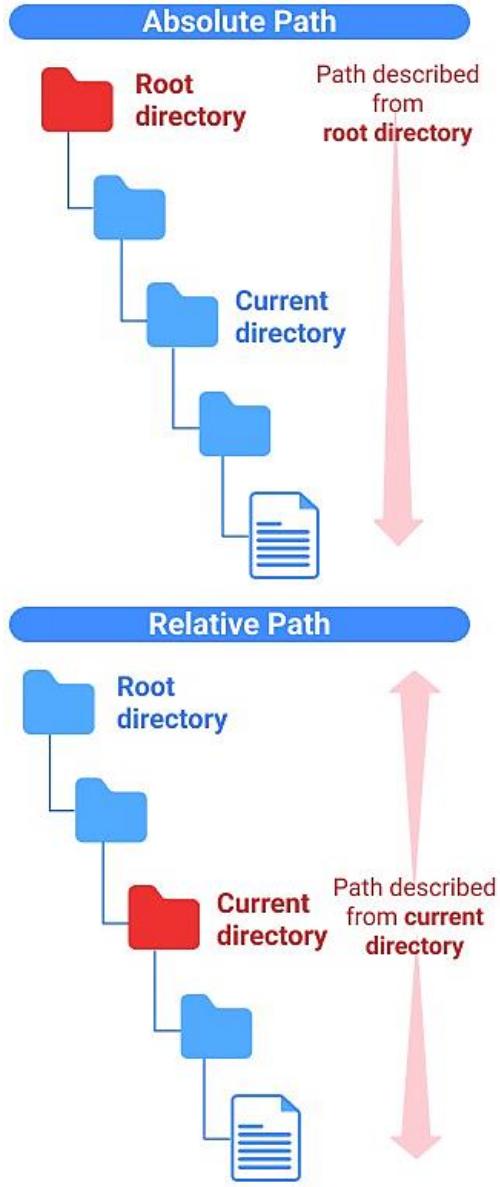
Absolute path :

An **absolute path** specifies the **complete location** of a file or directory starting from the **root directory (/)**.

It is also known as the **full path**, as it provides the entire hierarchy needed to locate the file.

- Always begins with a **forward slash (/)**.
- Remains **the same regardless of the current working directory**.
- Useful when the **file's location is fixed or when scripting**, to avoid ambiguity.

Example: `/home/student/Documents/report.txt`



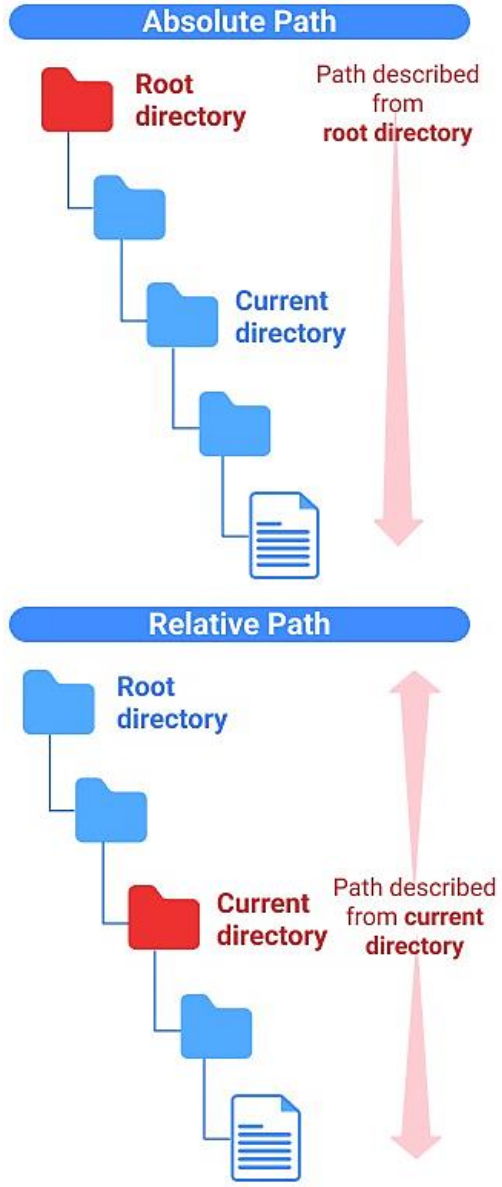
❑ File Access: Absolute Path and Relative Path

Relative path :

A **relative path** specifies the **location of a file or directory in relation to the current working directory**.
It does **not** start from the root (/), but from the directory you are currently in.

- Depends on the **current directory** (pwd).
- Uses **special symbols**:
- **.** → current directory
- **..** → parent directory
- Useful for **shorter, flexible paths** when working within project or user folders.

Example:
If you are in `/home/student`, then the **relative path** to `/home/student/Documents/report.txt` is:
`Documents/report.txt`



❑ File Access: Absolute Path and Relative Path

Relative path to a descendant directory or file :

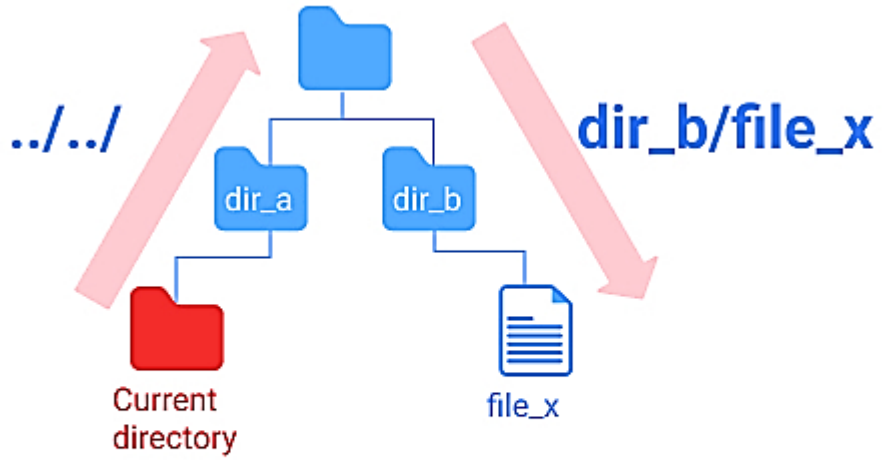
Pointing to a descendant directory or file is straightforward. You just need to describe the path to the file or descendant directory as follows:

```
directory name a / directory name b / file name
```

Relative path to an ancestor directory or file :

- ✓ To point to an ancestor directory or file, you need to use a **double-dot** (**..**).
- ✓ To describe a file path under the parent directory, you can write **`../file name`**.
- ✓ To go to a further higher directory, repeat **../**.

The illustration below shows the relative path from the current directory to a file under a different branch of the directory tree.



❑ File Access: Absolute Path vs. Relative Path

Feature	Absolute Path	Relative Path
Definition	Specifies the *complete location* of a file or directory starting from the **root (/)**.	Specifies the *location relative to the* ***current working directory***.
Starts With	**/** (root directory)	**No /**; starts from the current directory.
Dependency	Independent of the current directory.	Depends on the current working directory (**pwd**).
Stability	Fixed – always points to the same file or folder.	Variable – changes meaning depending on where you are.
Use Case	Used in scripts or configurations where file locations are constant.	Used for quick navigation and portable scripts within projects.
Example	`/home/student/Documents/report.txt`	`Documents/report.txt` or `../Documents/report.txt`

❖ FILES & DIRECTORIES

- All directories on the Linux system contain **two hidden directories**.
  - The **.** represents the **current** or **active directory**.
  - The **..** represents the **parent of the current directory.**
- A file or directory is **hidden** if its name begins with the **dot** (.)
- **Example**: Create a directory **Cours** and two other hidden directories **.TP** and **.TD** in the user1 desktop

```
user1@PC:~/Bureau$ pwd
/home/user1/Bureau
user1@PC:~/Bureau$ ls -a
.  ..  TD  TP
user1@PC:~/Bureau$ cd .
user1@PC:~/Bureau$ pwd
/home/user1/Bureau
user1@PC:~/Bureau$ cd ..
user1@PC:~$ pwd
/home/user1
```

```
user1@PC:~$ ls Bureau/
user1@PC:~$ mkdir Bureau/.TP Bureau/.TD
Bureau/Cours
user1@PC:~$ ls Bureau/
Cours
user1@PC:~$ ls -a Bureau/
.  ..  Cours  .TD  .TP
```

❑ Nomenclature of files and directories

- Linux OS distinguishes between **lowercase** and **uppercase characters**.

- **Examples :**
  - ✓ **File** and **file** two different names ;
  - ✓ "**TP1 G01**" is a directory with its name containing two spaces;
  - ✓ "**TP1 G01**" is a directory with its name containing a single space;
  - ✓ "**TP1*G01**" is a directory with its name containing *

```
user1@PC:~/Bureau$ mkdir TP1\ \ G01 'TP G01' TP\*G01
user1@PC:~/Bureau$ ls
'TP1  G01'  'TP G01'  'TP*G01'
user1@PC:~/Bureau$ ls 'TP*G01'
user1@PC:~/Bureau$ ls TP*G01
'TP1  G01':

'TP G01':

'TP*G01':
```

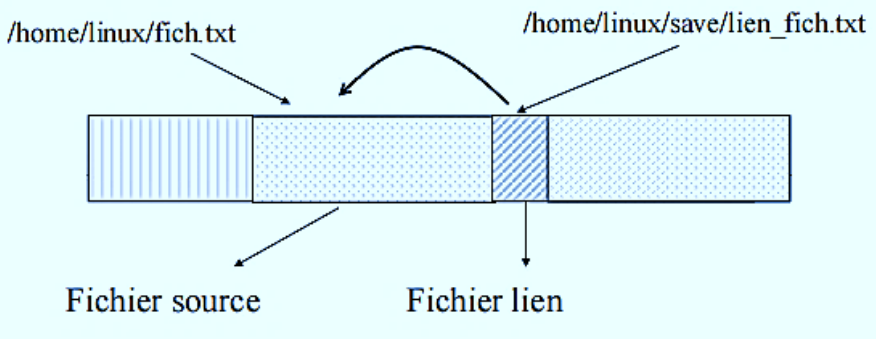
- **Notes on special characters :**
  - ✓ **Special characters** have a special meaning within the **shell** like: **~ <espace> ? * !**
  - ✓ To disable the interpretation of special characters, simply add a **backslash ** before each special character or enclose the string in **quotes "**.
  - ✓ It is recommended that you **avoid the use of special characters in file and directory names.**

❑ ln (Create Link to File and Directory)

The **ln** (**LiNk**) command is used to create links to files or directories. There are two types of links in Linux - **symbolic link** and **hard link**.

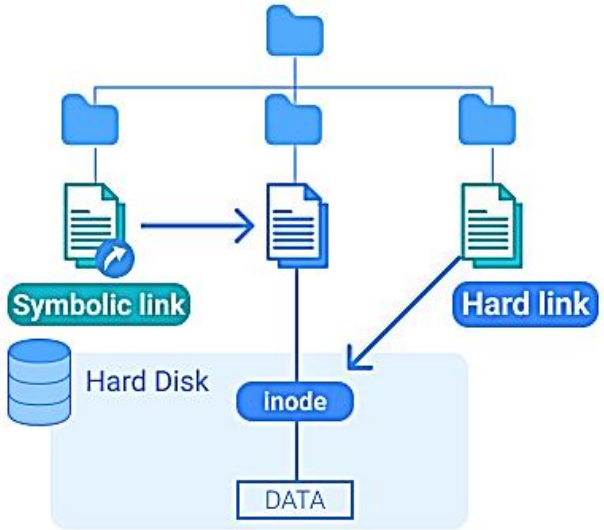
Symbolic Link:

The symbolic link is an alias on MacOS or a shortcut on MS Windows. It is linked with the original file or directory. Thus, when you delete the original file or directory, the symbolic link no longer works.



Symbolic link: `ln -s [file or directory path] [link name]`

Hard link: `ln [file or directory path] [link name]`



Symbolic link
- An alias or a shortcut of a file or directory
- When the original file is deleted, the symbolic link will no longer be working
- Symbolic links are applicable to both files and directories

Hard link
- A file has the same inode value as the original file
- Even when the original file is deleted, the hard link can still work by pointing to the same inode
- Hard links are only applicable to files.

Symbolic Link :

$ln -s Target Link_name

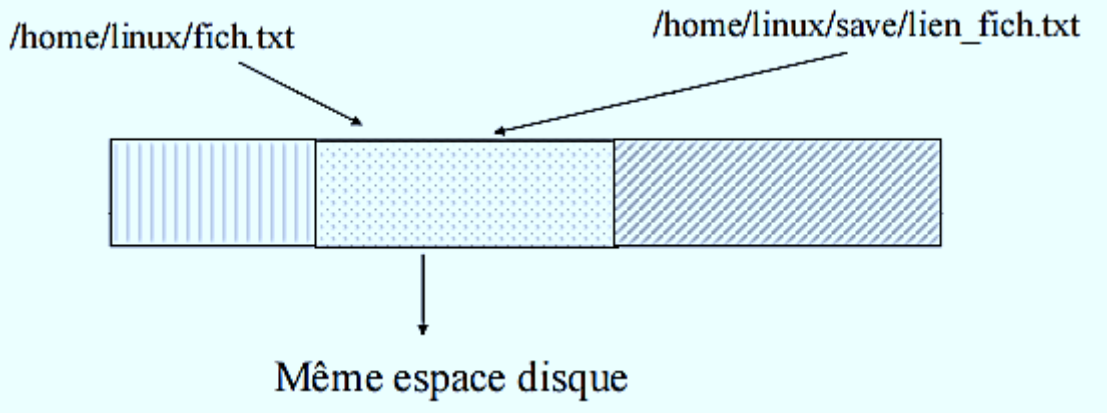
$ln -s /home/linux/linux.txt rep/linux1.txt

❑ ln (Create Link to File and Directory)

Hard Link:

The concept of the **hard link** is a bit more complex. In the Linux file system, the actual data of each file is stored in a certain location on the hard disk.
The location is described in the **inode** of each file.

When you create a hard link of **file_a**, the link is connected to the **inode** of **file_a** directly. Thus, even if you delete **file_a**, still the link can exist unless all the hard links that are linked with the **inode** are deleted and no processes are using the data.



Hard Link :
$ln file1.txt file1_hardlink.txt

❑ Special Characters and Escape Character

There are characters with special meanings assigned by **Linux OS** called **special characters**.

You cannot use the characters in a file name or normal text sentence. If you want to use them without having a special meaning in a sentence, you need to use an **escape character**.

In **bash**, \ (backslash) is used as an **escape character**.

Special Characters	Description
/	Path Directory Separator
.	Current Directory
~	Home Directory
#	Comment or Trim Strings
<>	Input and Output Redirection
|	Pipe
!	Pipeline logical NOT
()	Group Commands – subshell
{}	Group Commands – no subshell
$	Variable Expressions
&	Background Process
*	Character Sequence Wildcard
[]	Character Set Wildcard
?	Single Character Wildcard
;	Shell Command Separator

Escape Character in Bash

\

Backslash

e.g., \"ABC\" returns "ABC" in echo command

Special Characters to specify directories :

It is important to learn some special characters used for specifying certain directories.

/	.	..	~
slash	dot	double dot	tilde
Root Directory	**Working Directory**	**Parent Directory**	**Home Directory**

❖ SPECIAL CHARACTERS

- **~ (tilde)** is a special character that the **shell** will replace with the **absolute path** of the user's home directory (**/home/username**).

```
user1@PC:~$ echo  ~
/home/user1
user1@PC:~/Bureau$ ls  ~
Bureau  Cours  TD  TP
user1@PC:~/Bureau$ cd  ~
user1@PC:~$ ls
Bureau  Cours  TD  TP
```

- * is a special character that the **shell** will replace with **any possible string**, even the empty string.

```
user1@PC:~/Bureau$ tree ~/Bureau
/home/user1/Bureau
├── TD
│   ├── TD1
│   └── TD2
└── TP
    ├── TP1
    └── TP2
6 directories, 0 files
user1@PC:~/Bureau$ echo *
TD TP
user1@PC:~/Bureau$ echo */*
TD/TD1 TD/TD2 TP/TP1 TP/TP2
```

❖ SPECIAL CHARACTERS

- **?** is a special character that the **shell** will **replace with any possible character**.

```
user1@PC:~/Bureau$ tree ~/Bureau
/home/user1/Bureau
├── TD
│   ├── TD1
│   └── TD2
└── TP
    ├── TP1
    └── TP2
6 directories, 0 files
user1@PC:~/Bureau$ echo ?
?
user1@PC:~/Bureau$ echo ??
TD TP
user1@PC:~/Bureau$ ls T?/
TD/:
TD1  TD2
TP/:
TP1  TP2
```

- **!** is a special character that the **shell** will **replace with any possible string except the one after it!**

  Example : delete content from the directory **Bureau** except **Cours**

```
user1@PC:~$ ls Bureau/
Cours  TD  TP
user1@PC:~$ echo Bureau/!(Cours)
Bureau/TD Bureau/TP
user1@PC:~$ rm -rf Bureau/!(Cours)
user1@PC:~$ ls Bureau/
Cours
```

❑ What are the operations related to directories ?

- Create one or more directories;

```
$ mkdir [options] directory 1 [directory 2] ...[directory n]
```

`mkdir` [*directory path*]

-p option

create new directories with a tree structure

- Delete one or more directories;

```
$ rmdir [options] directory 1 [directory 2] ...[directory n]
$ rm -r [options] directory 1 [directory 2] ...[directory n]
```

`rmdir` [*directory path*]

Note:
- Cannot delete a directory if it is not empty
- Use the **rm** command with the **-r** option to delete a non-empty directory

- List (read) the contents of one or more directories;

```
$ ls [options] [directory 1] [directory 2]…[directory n]
```

`rm` [*file path*]

-r (recursive) option

delete a directory with contents (directories and files)

You cannot delete directories without the w (write) permission even if you are using the **-r** option.

- Change the active directory;

```
$ cd [directory]
```

❑ What are the operations related to directories ?

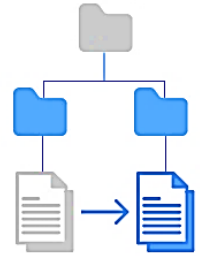
- Rename a directory;

```
$ mv [options] [directory source]
     [directory destination]
```

- Copy the contents of a directory;

```
$ cp -r [options] [directory source]
     [directory destination]
```

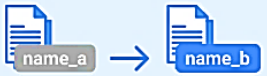
mv [*file or directory path*] [*destination path*]



Move a file or directory

Change existing file or directory name

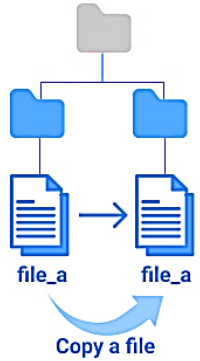
mv command is also used to change file or directory name.



mv [file or directory path] [new name]

Note:
Changing the file name (or directory name) is applicable only when there is no directory whose name is the same as the intended new file name. If such directory exists, the **mv** command moves the file (or directory) under it without changing the file's (or directory's) name.

cp [*source path*] [*destination path*]



Copy a file

Change the file name when creating a copy

use a new file name at the end of the destination path

cp [*source path*] [*destination path with new name*]

-r option : copy a directory

use the **-r** option to copy a directory

cp -r [*source path*] [*destination path*]

❑ What are the operations related to directories ?

- Know the location of the active directory;

```
$ pwd [options]
```

- Find the location of a directory;

```
$ find [options] [directory]
$ locate [options] [directory]
```

find [*path to start*] **-name** [*file or directory name*]

Search files or directories

Other Options	
`-type`	**specify file type** (e.g., f : normal file, d : directory, l : symbolic link, s: hard link)
`-size`	**search with size of document**
`-empty`	**search empty document**
`-o`	**or** (combine two search criteria)

Wildcard

Wildcard is useful when you forget the exact name of a file or directory.

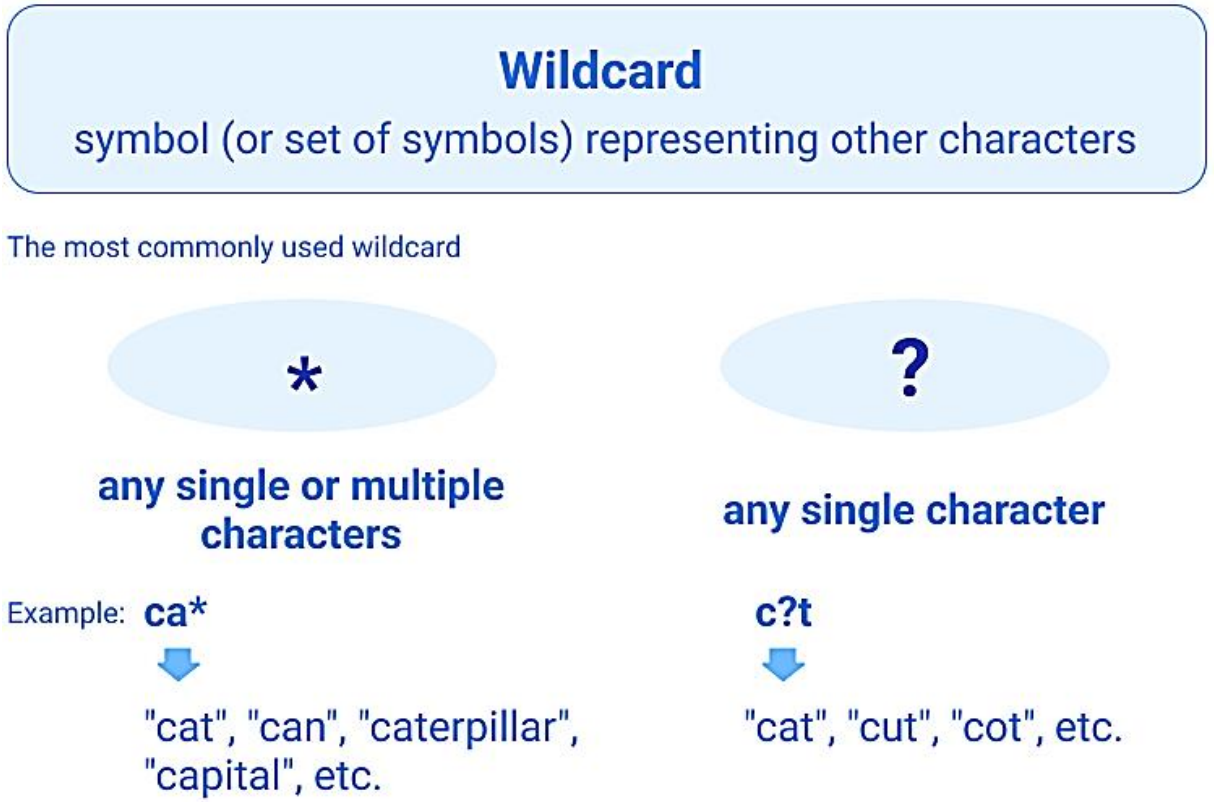
❑ Wildcard

Wildcard is a symbol (or a set of symbols) representing other characters. It is generally used for substituting any string or character.

Wildcard vs. Regular Expression

Wildcard and regular expression look similar but they are different things.

The definition of wildcard characters can differ by software while the definition of meta-characters in the regular expression is typically the same across different software.



❑ What are the file operations ?

- Create one or more files;

```
$ touch [options] file 1 [file 2] ... [file n]
```

- Delete one or more files;

```
$ rm [options] file 1 [file 2] ... [file n]
```

- Read the content of one or more files;

```
$ cat [options] [file 1] [file 2] ... file n]
$ more [options] file    $ less [options] file
```

- Rename a file;

```
$ mv [options] [file source] [file ou directory
                    destination]
```

- Find the location of a file;

```
$ find [options] [file]
$ locate [options] [file]
```

touch [*file name*]

Create a new file

Change existing file or directory timestamp

touch command is also used to change existing file or directory's timestamp (e.g., last modified date and time)

-d option
specify date and time

`touch -d [date and time] [file name]`

-r (--reference) option
apply another file's time stamp

`touch -r [reference file] [target file]`

cat [*file path*]

Display the content of a file

Create and edit a file

cat command is also used to create a new file or edit an existing file

1. Create a new file with text content

`cat > path of a new file`

2. Add text into an existing file

`cat >> path of an existing file`

3. Overwrite an existing file

`cat > path of an existing file`

❑ What are the file operations ?

- Find the location of a **binary file** (command);

```
$ which [options] commande name
$ whereis [options] commande name
```

- Copy a file;

```
$ cp [options] [file source] [file destination]
```

- Know the file type;

```
$ file [options] file
```

- Look for a word in a file;

```
$ grep [options] mot file
```

- Find and replace a word in a file;

```
$ sed -i 's/mot 1/mot 2/g' file
```

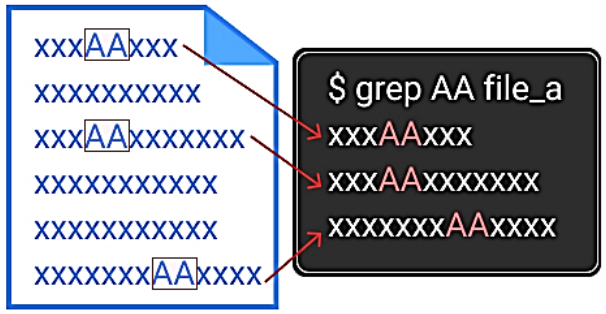
❑ What are the file operations ?

- **sort** (**Sort** File Contents)

- **grep** (**G**lobal **R**egular **E**xpression **P**rint)

grep [*search pattern*] [*file or directory path*]

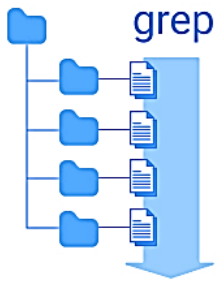
Phrase or regular expression



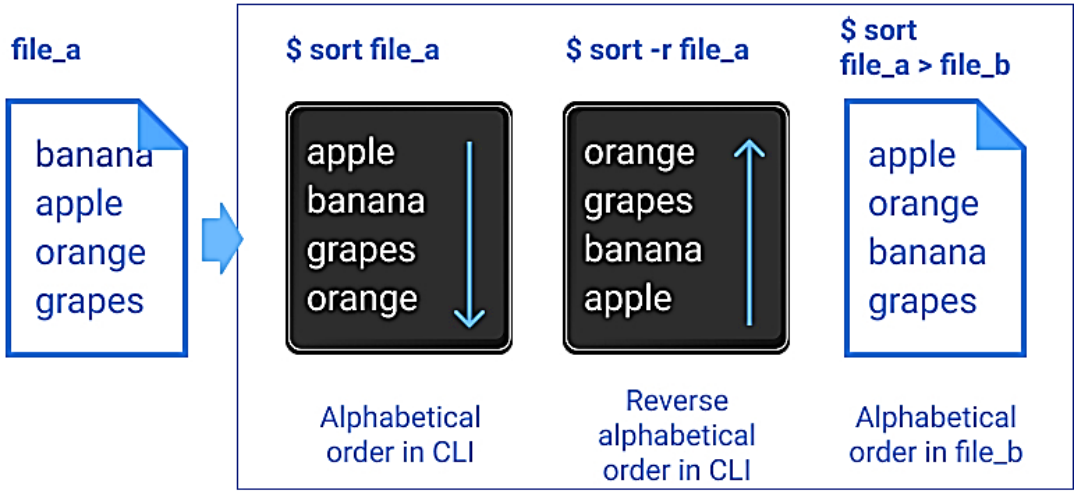
- Search for a string of characters (phrase) or a pattern in specified files
- Return the lines with the string

-r (recursive) option

search a string in multiple files under a specified directory



sort [*file path*]



❑ Regular Expression

Regular Expression is widely used in many programming languages. It is also called **regex** or **regexp**.

Basically, it defines the pattern of strings with meta characters to effectively identify specific data in a data set.

The **grep** command uses **regex** to specify **a search string pattern**.

Examples of meta-characters

Meta-Character	description
^	begin with
$	end with
[ ]	matches anything contained
[^]	matches anything NOT contained
.	matches any one character
*	character(s) specified before * or no occurrence

❑ tree (Display Directory Tree)

The **tree** command is used to display directories and files in a **tree structure** in your command line.

In the **CUI** environment, understanding the directory structure intuitively can be challenging. This command gives a solution for you.

Difference between ls and tree : When you run the **ls** command, you can see only one layer of a specified directory.
To understand the directory tree, you need to run the **ls** command multiple times; however, if you run the **tree** command, you can see the entire directory tree with one command.

tree [*directory path*] **-L** [*directory level*]

ls command

```
$ sudo ls /root
snap

$ sudo ls /root/snap
amazon-ssm-agent  lxd

$ sudo ls /root/snap/amazon-ssm-agent
2012  6312  common  current

$ sudo ls /root/snap/lxd
24164  24175  common  current
```

tree command

```
$ sudo tree /root
root
    snap
        amazon-ssm-agent
            2012
            6312
            common
            current -> 6312
        lxd
            24164
            24175
            common
            current -> 24175
```

What is the command that gives the size of a file or directory ?

The command **du** gives the size of one or more files or directories.

Syntax: `du [options] [fichiers] [répertoires]`

Some important options :

- **-h :** gives the size in human-readable format **K** (Kilo), **M** (Mega), **G** (Giga)
- **-s :** displays only the total for a directory.

Example: *Case of a file*

```
user1@PC:~$ du installation.MP4
265636   installation.MP4
user1@PC:~$ du -h installation.MP4
260M  installation.MP4
```

Example: *Case of a Directory*

with the **-s** option we will only have the size of the repertoire without the details.

```
user1@PC:~$ du -h -s  SYS1/
72K   SYS1/
user1@PC:~$ du -h SYS1/
4,0K  SYS1/TD
52K   SYS1/PROG
8,0K  SYS1/Cours
4,0K  SYS1/TP
72K   SYS1/
```

❑ How do I create a shortcut to a file or directory?

The **ln -s** command allows you to create a **symbolic link** (**shortcut**) to a file or directory.

Syntax : `ln -s file or directory shortcut`

Note: You must use **absolute paths** if you want to create a **shortcut** in a location other than the location of the file or directory.

Example: *(Shortcut to a file)*

```
user1@PC:~$ cat /etc/hostname
PC
user1@PC:~$ ln -s /etc/hostname /home/user1/raccourci_hostname
user1@PC:~$ cat raccourci_hostname
PC
```

Example: *(Shortcut to a directory)*

```
user1@PC:~$ ls /home/TP
G01  G02  G03  G04  G05  G06  G07  G08  G09  G10  G11  G12
user1@PC:~$ ln -s /home/TP   /home/user1/raccourci_TP
user1@PC:~$ ls raccourci_TP
G01  G02  G03  G04  G05  G06  G07  G08  G09  G10  G11  G12
```

How do I know the file type?

The command **file** is used to determine the type of one or more files or directories.

Syntax : `file [options] [fichiers] [répertoires]`

Example :

```
user1@PC:~$ file Bureau/ /etc/os-release /etc/hostname
/bin/cat
Bureau/:            directory
/etc/os-release:    symbolic link to ../usr/lib/os-release
/etc/hostname:      ASCII text
/bin/cat:           ELF 64-bit LSB shared object, x86-64,
version 1 (SYSV), dynamically linked, interpreter
/lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=b357ed53c8c9cb1a312f83b28982304effae0135,
for GNU/Linux 3.2.0, stripped
```

Search on a file or directory

The command **find** allows you to search for a file or directory.

Syntax : `find [options] [paths] [expression]`

- **paths :** the directories from which the search begins.
- **expression:** The name of the file or directory searched for.

Example :

A search on **TP1** from the **current directory .** (**/home/user1/Bureau**) and another from **parent directory ..** (**/home/user1/**)

```
user1@PC:~/Bureau$ tree ..
..
├── Bureau
│   └── TP
│       ├── TP1
│       └── TP2
└── TP
    ├── TP1
    └── TP2
7 directories, 0 files
user1@PC:~/Bureau$ find . -name "TP1"
./TP/TP1
user1@PC:~/Bureau$ find .. -name "TP1"
../Bureau/TP/TP1
../TP/TP1
```

❑ Other Useful Linux Commands

Linux provides many additional tools for searching, comparing, and managing files efficiently. Here are some **commonly used commands** you should know:

Search & Identification

- `locate` – Quickly finds files and directories by name (uses a prebuilt database).
- `which` – Displays the full path of a command or executable.

File Splitting & Merging

- `split` – Splits a large file into smaller parts.
- `join` – Joins lines from two files based on a common field.

File Comparison

- `diff` – Compares files **line by line** and shows differences (useful for code).
- `cmp` – Compares **two files byte by byte** to check if they are identical.

Archiving & Compression

- `tar` – Creates or extracts **archive files** (**.tar**, **.tar**, **.gz**, etc.).

💡 **Tip :**
Combine these commands with **pipes** (|) and redirection to build **powerful workflows** for system administration and scripting.

THANK YOU for your attention!

Questions ?

NSCS
المدرسة الوطنية العليا في الأمن السيبراني
NATIONAL SCHOOL OF CYBERSECURITY

مخبر هندسة الأنظمة المعقدة
LABORATOIRE D'INGÉNIERIE DES SYSTÈMES COMPLEXES

For more information about my research works, **Contact Information**:

Dr. Sassi BENTRAD

LISCO Laboratory : http://lisco.univ-annaba.dz/

☎ : +213 …

✉ : sassi.bentrad.enscs@gmail.com // sassi.bentrad@enscs.edu.dz

: www.linkedin.com/in/sassi-bentrad/

: http://www.bentrad-sassi.sitew.com/

ORCID
Connecting Research and Researchers

: orcid.org/0000-0002-7458-8121

RESEARCHERID : A-9442-2013

SCOPUS **Author ID** : 44461052600